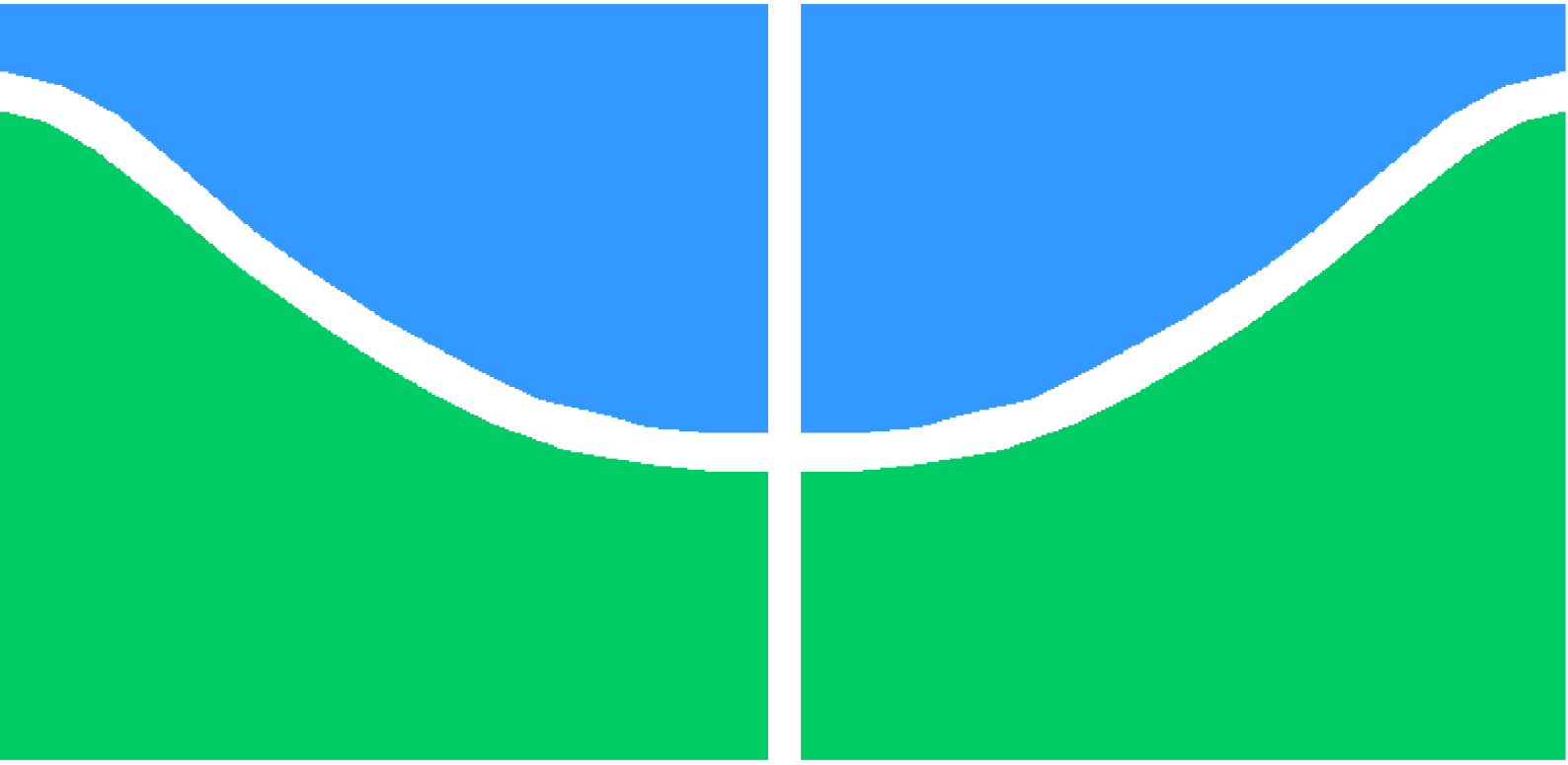

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Sistemas Legados: Uma Proposta para Centralização de Autenticação

Autor: Paulo Eduardo Souza Borba
Orientadora: Professora Dra. Edna Dias Canedo

Brasília, DF
2017

Paulo Eduardo Souza Borba

Sistemas Legados: Uma Proposta para Centralização de Autenticação

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF

2017

Paulo Eduardo Souza Borba
Sistemas Legados: Uma Proposta para Centralização de Autenticação/ Paulo Eduardo Souza Borba. – Brasília, DF, 2017-
78 p. : il. (algumas color.) ; 30 cm.

Orientador: Professora Dra. Edna Dias Canedo

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2017.

1. Autenticação. 2. SAML. 3. SOAP. 4. SOA. 5. SSO. I. Professora Dra. Edna Dias Canedo. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Sistemas Legados: Uma Proposta para Centralização de Autenticação

CDU 02:141:005.6

Paulo Eduardo Souza Borba

Sistemas Legados: Uma Proposta para Centralização de Autenticação

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 14 de Junho de 2017:

Professora Dra. Edna Dias Canedo
Orientador

Professor Ms.Ricardo Ajax Dias Kosloski
Convidado 1

Professor Ms.Giovanni Almeida Santos
Convidado 2

Brasília, DF
2017

Resumo

Com surgimento da era da informação, muitas empresas desenvolveram sistemas para aumentarem sua produção e crescimento, só que com o passar do tempo e o crescimento tecnológico, o sistemas que foram desenvolvidos naquela época se tornam ultrapassados, e com o tempo a necessidade de integração dos sistemas modernos e seus serviços, com os sistemas legados começaram a aumentar. A utilização de um barramento de serviços como o *Service Oriented Architecture*, vem para ajudar a integrar esses serviços em um meio de comunicação comum a todos os sistemas. Quando se pensa em integração de serviços, conceitos como autenticação e autorização são importantes para oferecer a segurança de identificação de usuários e políticas de acesso a serviços. A autenticação é um fator importante para integração, pois uma autenticação centralizada a todos os usuários e serviços, aumenta a segurança, diminui custos de manutenção e abaixa a complexidade de gerenciamento. A abordagem deste trabalho se dá no contexto de oferecer e explicar uma solução na autenticação única e centralizada de serviços no barramento SOA, com a utilização do protocolo de segurança chamado *Security Access Markup Language* (SAML) como solução desta modelagem.

Palavras-chaves: SAML; SSO; Autenticação; SOAP; SOA.

Abstract

Legacy systems, developed to meet the business rules of companies and their growing industries. With the advent of technological growth, companies develop new services to meet their business rules, and increasingly the need arises to integrate modern systems and their services, with legacy systems. The use of a service bus like the service-oriented architecture comes to help integrate these services into a common means of communication to all systems. When thinking about service integration, concepts such as authentication and authorization are important to provide user identification security and service access policies. Authentication is a key factor for integration because centralized authentication to all users and services increases security, lowers maintenance costs, and lowers management complexity. The approach of this work is in the context of offering and explaining a solution in the single and centralized authentication of services in the SOA bus, using the security protocol called Security Access Markup Language (SAML) as solution of this modeling.

Key-words: SAML; SSO; Authentication; SOAP; SOA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UnB	Universidade de Brasília
SOA	Service Oriented Architecture
CPD	Centro de Informática
SSO	Single Sign On
SAML2	Security Assertion Markup Language 2
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
HTTP	Hypertext Transfer Protocol
PL/I	Program Language One
SOC	Service Oriented Computing
UDDI	Universal Description, Discovery and Integration
SIGRA	Sistema de Informações e Gestão Acadêmica
XML	eXtensible Markup Language
RPC	Remote Procedure Call
W3C	World Wide Web Consortium
REST	Representational State Transfer
LDAP	Lightweight Directory Access Protocol
MSAD	Microsoft Active Directory
OASIS	Organization for the Advancement of Structures Information Standards
SSTC	Security Services Technical Commitee
PKI	Public Key Infrastructure
SSL	Secure Socket Layer
TSL	Transport Security Layer

IdP	Identity Provider
SP	Service Provider
URL	Uniform Resource Locator
JKS	Java Key Store
JSP	Java Server Page
CA	Certification Authority
CAS	Central Authorization Service
IDE	Integrated Development Environment
DWP	Dynamic Web Project

Sumário

1 Introdução

1.1 Contextualização

A tecnologia nos últimos anos tem evoluído cada vez mais, para atender as necessidades das empresas, respeitando o crescimento econômico. O crescimento globalizado impulsionou empresas a investirem no desenvolvimento de sistemas ao longo de gerações, para que possam acompanhar a tendência de crescimento do mercado. Porém, com o tempo, as empresas começaram a se acomodar em utilizar os sistemas legados como a base de todo um processo essencial para o funcionamento da empresa. Para Pinto e Braga (2004), os sistemas legados empregam valores críticos para o negócio de forma a automatizar a produção, diminuindo custos e erros, entretanto, o ponto essencial a se destacar é o tempo de utilização desses sistemas que muitas vezes chegam a 5, 10, 15, 20 anos de uso, com manutenções e evoluções em cima de plataformas antigas e ultrapassadas.

Com a defasagem dos sistemas legados ao longo dos anos, sua tecnologia vai se tornando obsoleta até chegar a um ponto em que será necessário desenvolver um novo sistema com base nas tecnologias atuais, e assim encerrando o ciclo de vida do sistema legado. As empresas aplicam manutenções e evoluções seguindo as tendências do ritmo de crescimento do mercado, para que possam acompanhar esse ritmo. Conforme é possível visualizar na figura 1, o ciclo de vida de um sistema legado em virtude do tempo, é alinhado com várias manutenções e evoluções para atender as mudanças das regras de negócio que vão evoluindo. Chega um momento em que após inúmeras iterações manutenções e evoluções, o sistemas não consegue acompanhar as tendências do mercado, devido às limitações tecnológicas, é nessa hora em que se decide pela substituição do sistema, para um sistema moderno que consiga acompanhar o ritmo de crescimento (WEIDERMAN et al., 1997).

Seguindo a linha de raciocínio sobre sistemas legados, muitas corporações, instituições e empresas, dispõem de um arsenal de sistemas para atender diversos departamentos com suas necessidades orientadas a inúmeras regras de negócio. Muitas vezes, sistemas independentes que precisam se comunicar com outros sistemas, passam por muitas dificuldades de integração pois precisam obter informações sobre determinado assunto para complementar o processamento dos dados que realizam, bem como usuários que precisam acessar múltiplos sistemas dentro de um mesmo domínio, porém, precisam de passar pela autenticação diversas vezes, devido a inexistência de integração entre os sistemas legados.

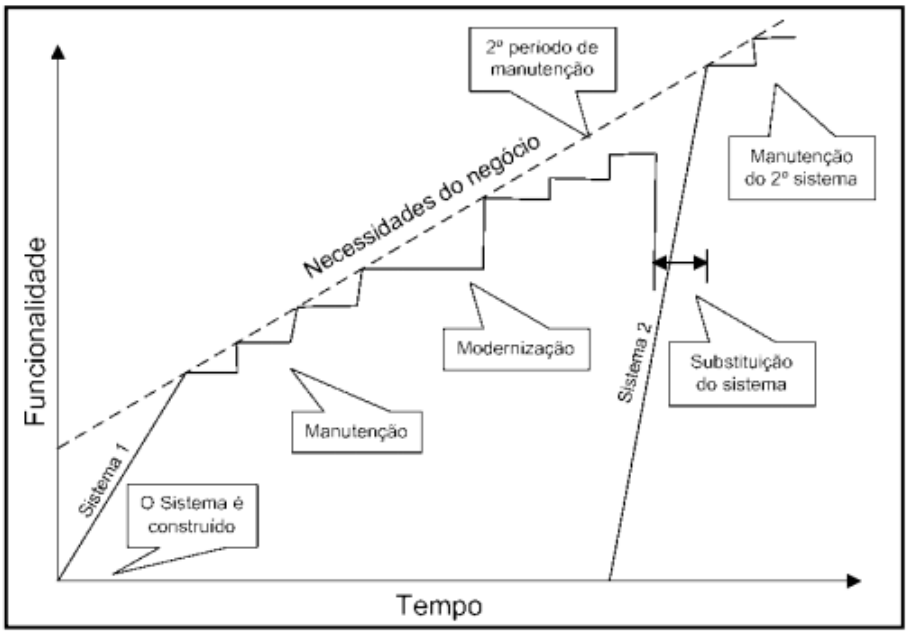

Figura 1 – Ciclo de vida de um sistema legado - Fonte: Pinto e Braga (2004)

Para isso, a melhor maneira de melhorar a experiência do usuário e dos serviços que necessitam se comunicar com outros serviços, é oferecer um barramento de serviços a qual todos os serviços da empresa(s) e ou instituição(ões) possam estar disponíveis em um só lugar. Contudo, um ponto importante a ser levantado é a questão de autenticação e autorização. Autenticação é forma a qual a aplicação identifica o usuário, e autorização é a política que define sob quais serviços, atributos, recursos e documentos este usuário tem acesso.

Sob o contexto geral definido, o Centro de Informática (CPD) da Universidade de Brasília (UnB), desenvolveu inúmeros serviços essenciais para o funcionamento interno da universidade. Apesar do funcionamento correto desses sistemas, muitas vezes existe a necessidade de um aluno ou servidor acessar múltiplos sistemas, seja para acessar a uma informação necessária que outro serviço requisita ou muitas vezes por que o serviço requisita de uma informação que não contém em seu sistema e se encontra em outro sistema determinado. De acordo com os problemas apresentados, o CPD desenvolveu um barramento para unificar os diversos sistemas legados da universidade em um só barramento, para facilitar a experiência do usuário, aumentando a segurança e diminuindo a complexidade de gerenciamento (AGILAR, 2016).

Definindo um contexto específico, este trabalho vai trabalhar sob um foco importante para garantir a integridade da identificação do usuário em um barramento de serviços. A integração proposta pelo CPD, será feita com base no *Service Oriented Architecture* (SOA) para transformar os sistemas legados em serviços a serem oferecidos e acessados pela universidade, de forma prática. Entretanto, o SOA não trata questões de

segurança como autenticação e autorização, e para isso este trabalho irá focar na autenticação do usuário a um barramento de serviços, de forma segura. Para isso, é proposto um sistema de autenticação centralizado de forma a garantir o compartilhamento da identificação do usuário entre os múltiplos serviços, com confidencialidade, autenticidade e de maneira integra.

1.2 Justificativa

Com base na existência de um barramento que integra os serviços da UnB, o CPD possui a preocupação sobre a necessidade de implantar mecanismos de segurança para garantir a integridade do sistema. O principal ponto de justificativa deste trabalho é relacionado a autenticação dos usuários, para isso é necessário aborda um mecanismo de autenticação e testa-lo para ver se atende ao barramento, conforme contextualizado por Agilar (2016).

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo deste trabalho é apresentar uma solução a autenticação única e segura de forma centralizada, utilizando o conceito de *Single Sign On* (SSO), com a utilização do protocolo de autenticação *Security Assertion Markup Language*.

1.3.2 Objetivos Específicos

Para cumprir o objetivo geral, é apresentado os seguintes objetivos específicos definidos:

- Definir um modelo de autenticação centralizada;
- Apresentar o protocolo de autenticação que será utilizado;
- Implementar pseudo serviços de simulação que irão simular a implantação de serviços em barramento;
- Montar um ambiente de teste e definir cenários de teste;
- Realizar os testes e colher os resultados;
- Realizar análise dos resultados obtidos bem como a conclusão que foi obtida com os testes realizados;

1.4 Questões de Pesquisa

Este trabalho procura responder as seguintes perguntas: A Modelagem Centralizada de Autenticação atende as necessidades de segurança na autenticação em um barramento de serviços? Com essa modelagem, pode se garantir a integridade de identidade do usuário e ou serviço em um barramento após sua autenticação?

1.5 Metodologia

A metodologia escolhida para a abordagem deste trabalho será uma pesquisa exploratória e investigativa. A pesquisa exploratória realiza uma abordagem de se familiarizar com o fenômeno, e junto a uma abordagem investigativa, a pesquisa exploratória traz benefícios de entender as variáveis que regem a ação do fenômeno e seu comportamento (CARMO; FERREIRA, 2008). Esta será uma abordagem qualitativa, com vista em realização de procedimentos técnicos embasados na realização de testes, juntamente com auxílio de pesquisas bibliográficas.

Com base na metodologia definida, foi modelado um processo que visa uma melhor abordagem da exploração e investigação de forma metódica. Utilização de testes, abordagem de um modelo centralizado de autenticação serão os objetos de estudo e análise sob a supervisão deste processo modelado, conforme pode ser visualizado na figura 2. E os seguintes pontos podem ser resumidos em:

- Definir o contexto de aplicação - Será apresentado um contexto atrelado junto a questão de pesquisa;
- Explicar tecnologias envolvidas - Apresentar fundamentação teórica de acordo com as tecnologias que estão sendo utilizadas;
- Apresentar modelo centralizado de Autenticação - Explicação do modelo centralizado que está sendo abordado;
- Implementar pseudo serviços - Implementação de pseudo serviços para imitar um barramento de serviços;
- Preparar ambiente de testes - Preparar o ambiente do modelo centralizado para realizar os testes devidos;
- Definir Testes Funcionais e realiza-los - Definir os cenários de testes e executa-los;
- Colher Resultados - Realizar a coleta dos resultados e apresentá-los;
- Realizar Conclusão - Apresentar a conclusão de acordo com os resultados obtidos e se o trabalho atingiu os objetivos desejados;

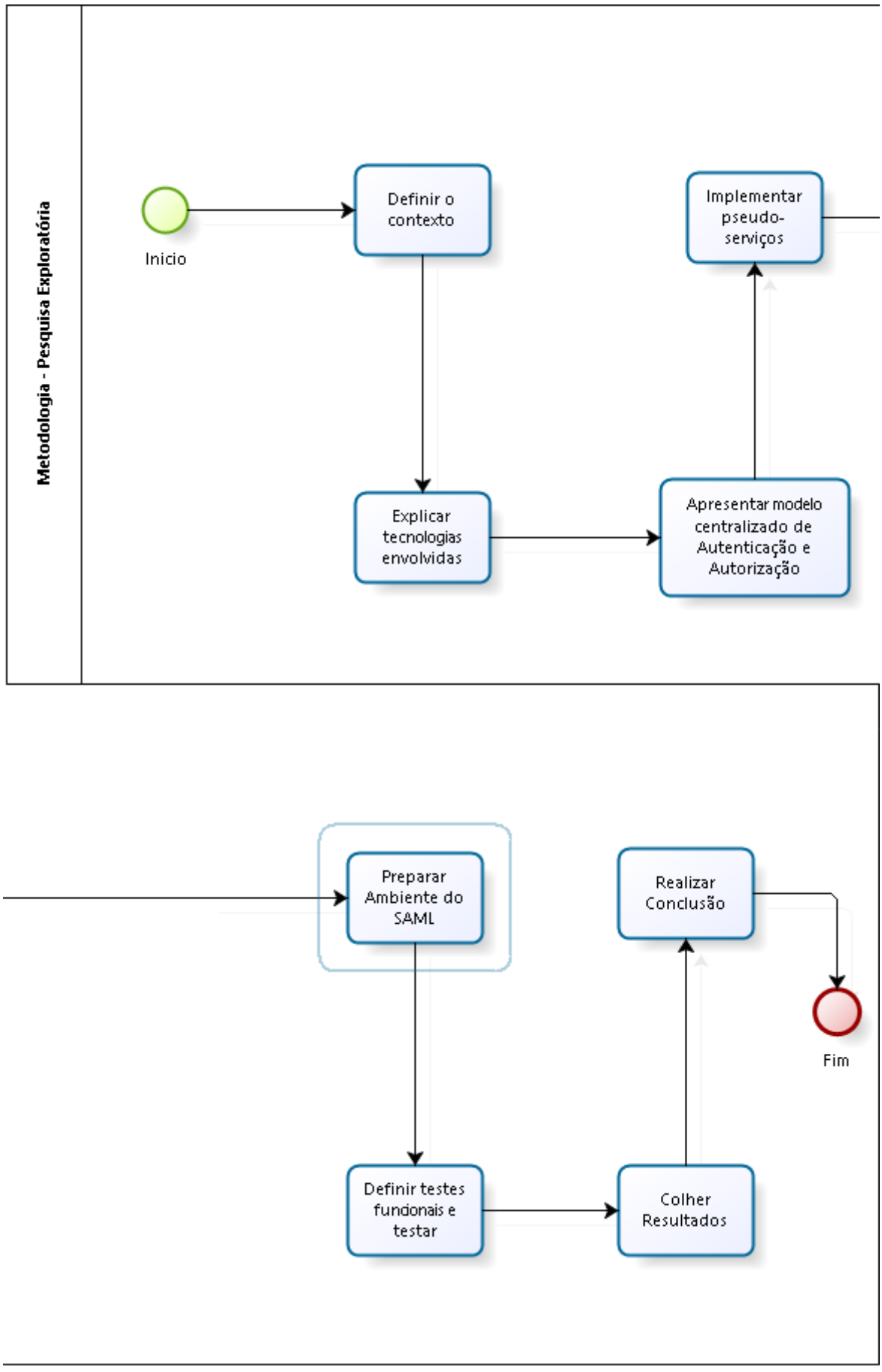

Figura 2 – Processo de Metodologia adotado para estudo investigativo - Fonte: Autor

1.6 Contribuição

A contribuição deste trabalho de conclusão de curso é a abordagem em uma modelagem centralizada de autenticação a ser oferecida, com o objetivo de aumentar a segurança da autenticação e permitindo a integração de sistemas, bem como diminuindo custos de manutenção e baixando a complexidade de gerenciamento.

O ecossistema de autenticação da WSO2 ajuda a viabilizar a integração da autenticação por *Single Sign On* (SSO) via SAML, pois devido à grande dificuldade de entendimento do protocolo, bem como a manipulação de seus dados e metadados, a implementação do WSO2 oferece um arcabouço com inúmeras ferramentas, que ajudam aos desenvolvedores montarem uma solução de fácil integração, entendimento e manutenção. Com isso temos a garantia de uma autenticação segura em que todos os serviços e usuários são identificados no barramento.

1.7 Organização do Trabalho

Este trabalho contém fundamentação teórica sobre SOA, SAML2 e SOAP, com o objetivo de apresentar os principais pontos que compõe o modelo de autenticação centralizada. Em seguida é apresentado o desenvolvimento da implementação dos pseudo-serviços utilizados para simular multiplas aplicações, e com isso o funcionamento do modelo apresentado. No capítulo 2 é exposto os conceitos utilizados e trabalhos correlatos. No capítulo 3 é apresentado o modelo de autenticação centralizado, e a implementação do sistema de autenticação com serviços simulatos. Por fim, é apresentado no capítulo 4 um breve resumo do trabalho realizado, bem como a resposta para a questão de pesquisa, e por fim trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta uma abordagem em diversos conceitos relacionados ao tema do trabalho, no caso da conceituação de sistemas legados com foco na integração através da arquitetura orientada a serviços. O Capítulo está dividido da seguinte forma: a Seção 2.1 relata de forma breve o que seria um sistema legado, e como se comporta o ciclo de vida geral de um sistema legado. Na Seção 2.2 é apresentado uma breve explicação sobre o funcionamento de um *Service Oriented Architecture* (SOA), incluindo seus componentes e o fluxo de dados. Na Subseção 2.2.1 é apresentado a estrutura de funcionamento do *Simple Object Access Protocol* (SOAP), o seu fluxo de dados de acordo com uma requisição pelo *Hypertext Transfer Protocol* (HTTP), e os componentes que o compõe. Na Seção 2.3 é apresentado o breve conceito de autenticação, e na Subseção 2.3.1 é explicado o protocolo de autenticação *Security Authentication Markup Language* (SAML), sua estrutura, regras, requisitos para que possa ser utilizado. Por fim, na Seção 2.4, é apresentado a justificativa de escolha do SOAP e do SAML.

2.1 Sistemas Legados

Sistemas legados tipicamente são a espinha dorsal do fluxo de dados de uma organização e o principal veículo de consolidação de informações de negócio. Eles são cruciais, e caso venham a apresentar falhas, eles podem gerar enormes impactos na empresa em que são utilizados (BISBAL et al., 1999). Pode-se dizer que a definição de um sistema legado vem como "qualquer sistema da informação que resiste significativamente a qualquer modificação e evolução ao longo do tempo", e essa resistência pode trazer enormes prejuízos para as organizações que o utilizam como: tecnologia e ou hardware ficar obsoleto com o tempo, alto custo de manutenção, documentação defasada, interface não amigável e difícil extensão e integração com outros sistemas (BISBAL et al., 1999).

O ciclo de vida de um sistema legado pode ser composto pelas seguintes fases:

- **Concepção:** O sistema é concebido para atender as regras negócio e tentar automatizar e melhorar a produtividade dado um problema especifico;
- **Manutenção:** Nesta fase o sistema legado passa por diversas iterações para que possa atender a mudanças nas regras de negócio ao longo do tempo;
- **Restruturação:** As manutenções se tornam cada vez mais custosas e não atendem a evolução das regras de negócio, o sistema legado passa por uma avaliação e res-

truturação para colocar novas tecnologias integradas para que possam atender as necessidades envolvidas;

- **Substituição:** Nesta fase, um novo sistema é refeito com base nos serviços oferecidos pelo sistema legado, porém acompanhando as novas tecnologias de software e hardware, para atender as demandas e seguindo as novas tendências tecnológicas;

Este modelo proposto por Bisbal et al. (1999) e por Comella-Dorda et al. (2000) e adaptado por Pinto e Braga (2004), além de apresentarem este ciclo de vida, propõem diversas técnicas de integração entre sistemas legados e sistemas modernos.

Vale destacar que segundo Sneed (2006), os sistemas legados também podem ser divididos em três básicas categorias também, dependendo de seu ambiente, são eles:

- Sistemas legados que não são dependentes do ambiente;
- Sistemas legados que são parcialmente dependentes do ambiente;
- Sistemas legados que são totalmente dependentes do ambiente;

O ambiente neste contexto se refere ao contexto de interpretação do código do sistema legado, por exemplo: linguagens como Fortran, COBOL e C/C++ possuem compiladores que podem ser usados em outros sistemas, bastando instalar o compilador para executar as rotinas do sistema legado. Diferente da categoria de sistemas que dependem parcialmente ou totalmente do ambiente, esses sistemas devem ser tratados com mais cautela em uma devida situação de Manutenção e ou Restruturação, por exemplo: no caso de sistemas que pertencem a *Program Language One* (PL/I), *SmallTalk* e *Forté*, eles podem ser usados em outro ambiente, mas somente se suas rotinas executáveis forem substituídas por módulos escritos compilados na linguagem da máquina hospedeira. E por fim, no caso de sistemas totalmente dependentes é necessário um ambiente especifico ou adaptado, para que atenda aos requisitos do sistema, como no caso de sistemas que usam *ADS-Online*, *Natural*, CSP e *Oracle Frames* (SNEED, 2006).

2.2 Service Oriented Architecture (SOA)

Service Oriented Computing (SOC) é o paradigma da computação que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações e soluções. A construção de um modelo de serviço recai no SOA que é uma forma de reorganizar as aplicações de software e infraestrutura em um conjunto de serviços interativos. Entretanto, o básico de SOA não retrata preocupações como gerenciamento, orquestração de serviços, gerenciamento de transações e coordenação de serviços, segurança e outros tópicos que se aplicam a todos os componentes da arquitetura de serviços (PAPAZOGLOU, 2003).

De forma resumida por Agilar (2016), SOA retrata uma arquitetura em que os serviços são criados, reutilizados e compartilhados entre os diversos sistemas de uma ou mais organizações, dentro de um sistema distribuído de modo que suas funcionalidades estão disponíveis aos interessados que desejam utiliza-las.

Serviços podem ser componentes encapsulados de software que retratam a regra de negócio em um alto nível, e normalmente são compostos pelos seguintes elementos: interface, contrato e o serviço consolidado (AGILAR, 2016). A interface é a ponte de acesso entre o fornecedor do serviço e o cliente, onde será viabilizada as requisições realizadas; o contrato é referente as regras de funcionalidade do serviço, parâmetro e tipo de parâmetros permitidos, restrições e dentre outras regras; e por fim o serviço consolidado é a implementação do serviço em código (KRAFZIG et al., 2004).

Os serviços oferecidos são conectados a um barramento juntamente com outras aplicações prontas para oferecer o serviço, que é o caso das aplicações de interface. Entretanto, os serviços podem ser oferecidos em diferentes plataformas e ou com utilização de diferentes protocolos de comunicação pela internet, e para atender aos requisitos de um serviço a qual todos podem acessar independente da heterogeneidade dos sistemas, os seguintes requisitos foram definidos:

- **Tecnologia Neutra:** Eles precisam ser invocados pelo menor denominador tecnológico comum e padrão que esteja disponível a todos os ambientes dos sistemas de informação. Isso implica em que os mecanismos de invocação como protocolos, descrições, e mecanismos de descoberta precisam atender aos padrões amplamente aceitos.
- **Baixo Acoplamento:** Não precisam de conhecimento ou estrutura ou convenções tanto no lado do cliente ou do serviço.
- **Suporte a transparência de localidade:** Serviços devem ter suas definições e informações de localidade armazenados em um repositório como o *Universal Description, Discovery and Integration* (UDDI). Os clientes poderão acessas essas informações e invocar os serviços nas suas respectivas localidades.

Com essa abordagem conceitual, feita por Krafzig et al. (2004), de acordo com Agilar (2016), que retrata a abordagem de SOA nos sistemas da UnB, é possível afirmar que a utilização de SOA representa uma estratégia para a modernização do Sistema de Informações e Gestão Acadêmica (SIGRA) da UnB, dados seus benefícios de integração com outros sistemas.

Os componentes do SOA podem ser resumidos em: Provedor de Serviço, que disponibiliza acesso aos serviços e permite também aos clientes localizarem os serviços e

executar a requisição deste serviço no provedor; Consumidor, trata-se dos usuários e ou aplicações que utilizam os serviços; Catálogo de Serviços, responsável pela localização central dos serviços, bem como em qual lugar o provedor pode publicar os serviços para que o consumidor utilize-os (AGILAR, 2016).

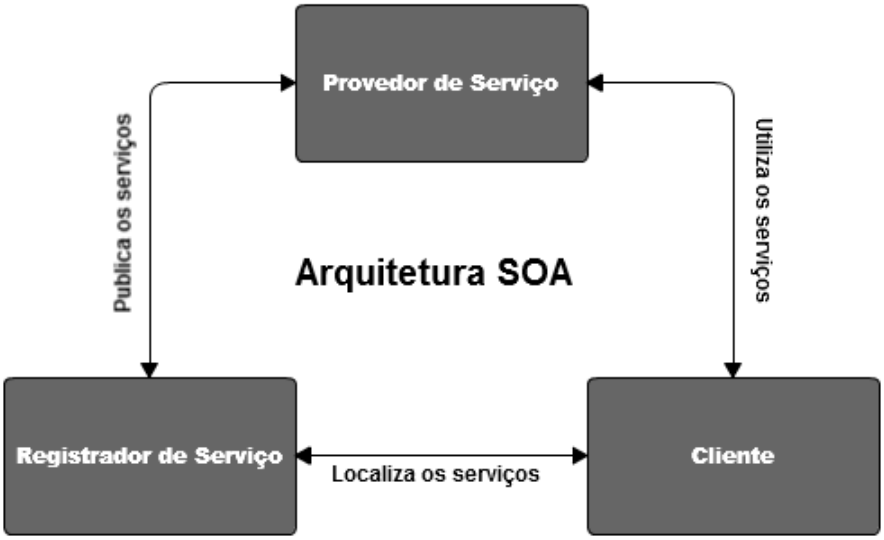

Figura 3 – Arquitetura básica do SOA - Fonte: Krafzig et al. (2004)

2.2.1 Simple Object Access Protocol (SOAP)

O *Simple Object Access Protocol* (SOAP) é um protocolo leve de troca de informações em um ambiente distribuído, descentralizado. É um protocolo baseado em *eXtensible Markup Language* (XML) que consiste nas seguintes partes: o envelope que define a descrição sobre o conteúdo da mensagem e como processa-la; um conjunto de regras codificadas para expressar instancias da aplicação definida em tipos de dados; e uma convenção para representar chamadas de procedimento remoto e respostas (BOX et al., 2000).

O SOAP é uma recomendação da World Wide Web Consortium (W3C), significa que o SOAP segue uma série de padrões que foram desenvolvidos em consenso para atender a todos os sistemas. Existem dois tipos de requisição SOAP: A primeira requisição é chamada de Remote Procedure Call (RPC), similar a arquitetura distribuída, em que o cliente envia uma mensagem e aguarda pro uma mensagem de erro ou resposta do servidor; o segundo tipo é uma requisição de documento, neste caso, um documento XML é repassado do cliente para o servidor dentro da mensagem SOAP (SUDA, 2003).

Para este trabalho será usado SOAP com a formatação em XML. Nesse estilo de mensagem, o cliente e servidor repassam mensagem em formatação XML utilizando o corpo da mensagem SOAP ao invés de parâmetros. Esse estilo é utilizado pelas aplicações em Web Services, que garante um processamento assíncrono e pode ser colocado em uma

fila. A confiança da aplicação é melhorada pois a mensagem em fila garante-se que será entregue mesmo se a aplicação alvo não estiver ativa (SUDA, 2003).

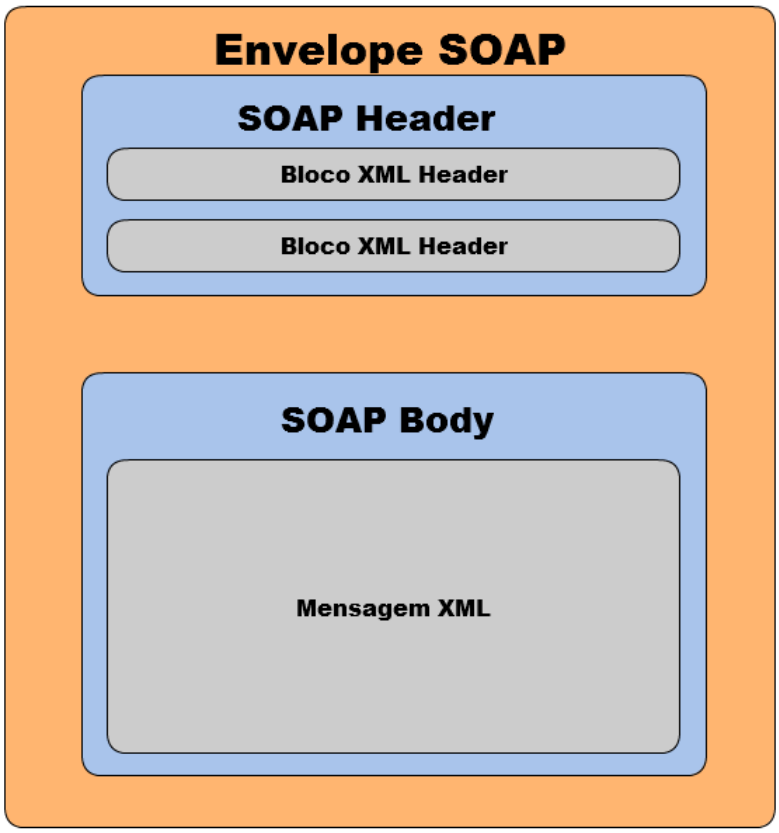

Figura 4 – Diagrama de uma mensagem SOAP - Fonte: Suda (2003)

Tabela 1 – Exemplo de uma mensagem SOAP sob contexto - Fonte: Curbera et al. (2002)

```
POST /travelservice
SOAPAction: "http://www.acme-travel.com/checkin"
Content-Type: text/xml: charset="utf-8"
Content-Length: nnnn

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelopse/">
    <SOAP:Body>
        <et:eTicket xmlns:et="http://schemas.xmlsoap.org/soap/envelope">
            <et:passengerName first="Joe" last="Smith"/>
            <et:flightInfo airlineName="AA"
                           flightNumber="1111"
                           departureData="2002-01-01"
                           departureTime="1905"/>
        </et:eTicket>
    </SOAP:Body>
</SOAP:Envelope>
```

O elemento raiz é a *tag* envelope, que contém os elementos de *body* e *header* como

é possível ver na imagem 4. O *body* provê um mecanismo para troca mandatória de informações intencionada para o remetente, ele contém elementos em XML de forma estruturada para o retorno de dados ou argumentos, ou apresentação de algum mecanismo de erro. Já a *header* é uma forma de aplicar propriedades de forma customizada para as aplicações (SUDA, 2003). A tabela 1 apresenta um exemplo de como seria uma mensagem SOAP no contexto de uma companhia aérea, esse contexto foi apresentado por Curbera et al. (2002), para demonstrar os diferentes aspectos e adaptações que SOAP pode ter.

2.2.2 Representational State Transfer (REST)

Outro protocolo comumente usado pelo Web Service é o *Representational State Transfer* (REST), em que seu estilo de arquitetura é cliente/servidor, o cliente envia a requisição para o servidor e então o servidor processa a requisição e envia a resposta. Essas requisições e respostas são construídas entorno da transferência da representação dos recursos. O recurso para o REST é identificado pela *Uniform Resource Identifier* (URI), e neste caso a representação do recurso é tratada como um documento que captura o estado corrente ou intencionado (MUMBAIKAR; PADIYA et al., 2013).

Diferentemente do SOAP, REST não utiliza um formato de mensagem baseado em body e header. O princípio do design do REST segue as seguintes características:

- ***Stateless:*** Baseado na interação entre cliente e o servidor em que não se mantem estados entre a troca de dados. Ou seja, as requisições do cliente para o servidor contêm as informações que descrevem a requisição, uma vez que o servidor não possui meios de armazenar a sessão e ou interpretar sem o devido contexto;
- ***Addressability:*** Baseado no fato em que todo o recurso possui uma URI, ele é acessível se enviado a mensagem contextualizando a requisição para o servidor;
- ***Uniform Interface:*** O acesso a todos os endereços segue uma interface uniforme, independente de heterogeneidade do serviço;

Essas características que foram definidas segundo Mumbaikar, Padiya et al. (2013), foram utilizadas para desenvolver o RESTful baseado em manter em cache os estados das aplicações entre cliente e servidor, e este protocolo utiliza os métodos HTTP de GET, POST, PUT, DELETE para recuperar, criar, atualizar e remover recursos (AGILAR, 2016). A representação mais comum dos dados que o RESTful utiliza é o JavaScript Object Notation (JSON) que é uma notação para representação de dados bem mais simpleficada que o XML. Na figura 5 é possível visualizar a utilização dos métodos para realizar a troca de mensagens em seu devido contexto.

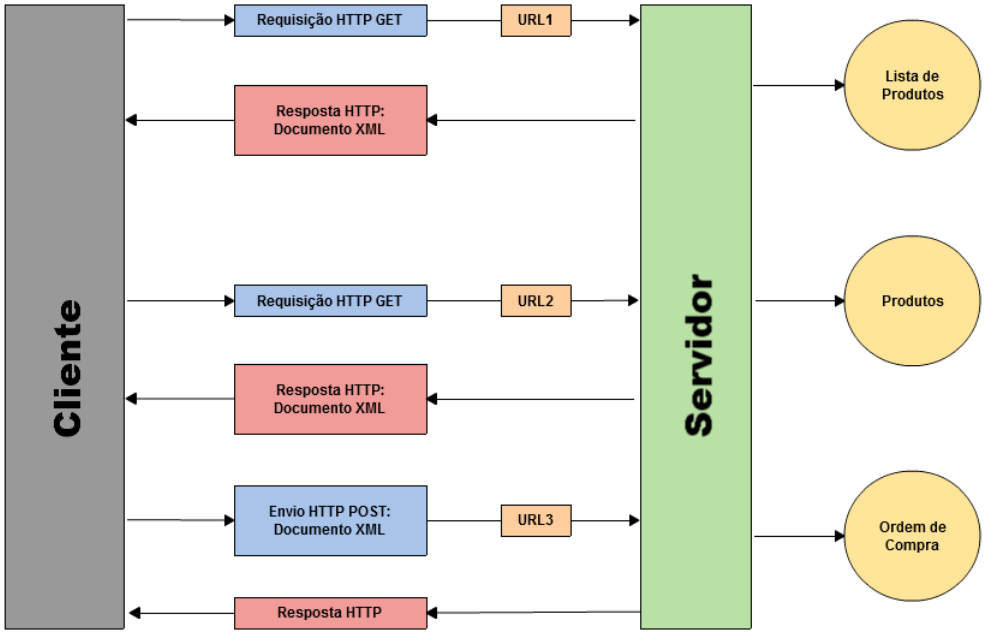

Figura 5 – Exemplo de troca de dados via REST - Fonte: Mumbaikar, Padiya et al. (2013)

2.3 Autenticação

Hoje em dia as aplicações possuem uma forma de autenticação para que os usuários possam acessar os recursos a qual desejam. Antes de qualquer autorização possa ocorrer, a aplicação precisa identificar o usuário, então a maioria das aplicações contam com pelo menos uma forma de autenticação em diversos tipos (LOPEZ; OPPLIGER; PERNUL, 2004). Alguns exemplos de autenticação são:

- *Lightweight Directory Access Protocol* (LDAP);
- *Microsoft Active Directory* (MSAD);
- Banco de dados customizado para guarda informações de login e senha do usuário;

Em todos esses aspectos, o usuário é identificado juntamente com seu perfil e mantido nas diferentes formas de aplicação. No contexto oferecido a autenticação, é o principal foco deste trabalho, e nesse caso será a primeira camada de serviço para que o usuário possa acessar os diferentes serviços oferecidos pela UnB.

2.3.1 Security Assertion Markup Language (SAML)

O barramento SOA de *Web Service* é um barramento desenvolvido para que os usuários possam acessar diferentes serviços independentemente da plataforma. Entretanto, uma grande preocupação está em saber se o usuário possui o acesso correto a certos serviços de acordo com o seu perfil e se ele passou por um processo de identificação que garanta

a integridade do usuário. Para isso será explicado o conceito do *Security Access Markup Language* (SAML) que é um protocolo de segurança para autenticação desenvolvido sob os padrões da *Organization for the Advancement of Structures Information Standards* (OASIS) ou organização para o avanço de padrões de estrutura de informações.

Os padrões do SAML definem um arcabouço para a troca de informações de forma segura entre as aplicações de negócio, mais precisamente ele define um arcabouço comum de XML para troca de asserções de segurança entre entidades, de forma portátil em que essa troca possa cruzar as informações entre os limites do domínio que são confiáveis (RAGOUZIS et al., 2008).

O SAML foi desenvolvido para atender uma série de critérios de segurança na troca de informações, dentre as características do SAML as principais são:

- ***Single Sign-On (SSO):*** A autenticação hoje em dia é baseada em *Cookies* que armazena o estado de sua autenticação em determinado serviço sem ter que passar pela autenticação novamente. Ou seja, seu estado de autenticado é mantido nas preferências de usuário, sem ter a necessidade de se autenticar, bastando apresentar o estado de autenticado para o provedor de serviço para acessar os serviços novamente, sendo assim autenticação única;
- ***Federated identity:*** O conceito de identidade federada é quando múltiplos serviços entram em acordo sobre como referenciar os usuários. De uma perspectiva administrativa, esse tipo de acordo de entendimento pode ajudar a reduzir os custos de gerenciamento de identidade, assim como múltiplos serviços não precisam coletar e manter dados da identidade relacionada. Com relação aos administradores desses serviços não precisam manter e estabelecer identificadores compartilhados, o controle disso pode residir no usuário;
- ***Web Service* e outros padrões industriais:** O SAML permite também ser personalizado para contexto de outros padrões industriais e ou Web Services customizados;

Para o contexto de centralização de autenticação, será utilizado o SSO, para prover integridade do usuário entre os serviços e garantir que o usuário seja integro e que tenha acesso aos serviços disponibilizados.

Arquitetura SAML

O SAML consiste na construção de blocos de componentes, quando colocados juntos em uma determinada ordem, permite seu uso aos diferentes casos de uso. Os componentes primariamente permitem transferir a identidade, autenticação, atributos, e informações de autenticação entre os serviços autônomos que estabeleceram uma relação

de confiança. O núcleo do SAML define a estrutura e o conteúdo de ambas asserções e protocolo de mensagens usados para transferir informações (RAGOUZIS et al., 2008).

Segundo Ragouzis et al. (2008), as asserções do SAML são "declarações" em forma de afirmações, então uma parte envia requisições para afirmar que suas "declarações" são verdadeiras e o SAML valida se são ou não. A estrutura valida e o conteúdo das asserções são definidos pelo SAML em XML. As asserções são usualmente criadas requisições de algum serviço ou usuário, assim que enviadas, elas são reenviadas ao alvo da requisição de forma não solicitada indicando o resultado das asserções

O SAML *Binding* é definido como o meio de transporte de mensagens do protocolo SAML em baixo nível como SOAP e é usado definindo o seu padrão de *Tags* para definir o meio de transporte entre os serviços e ou usuários (RAGOUZIS et al., 2008).

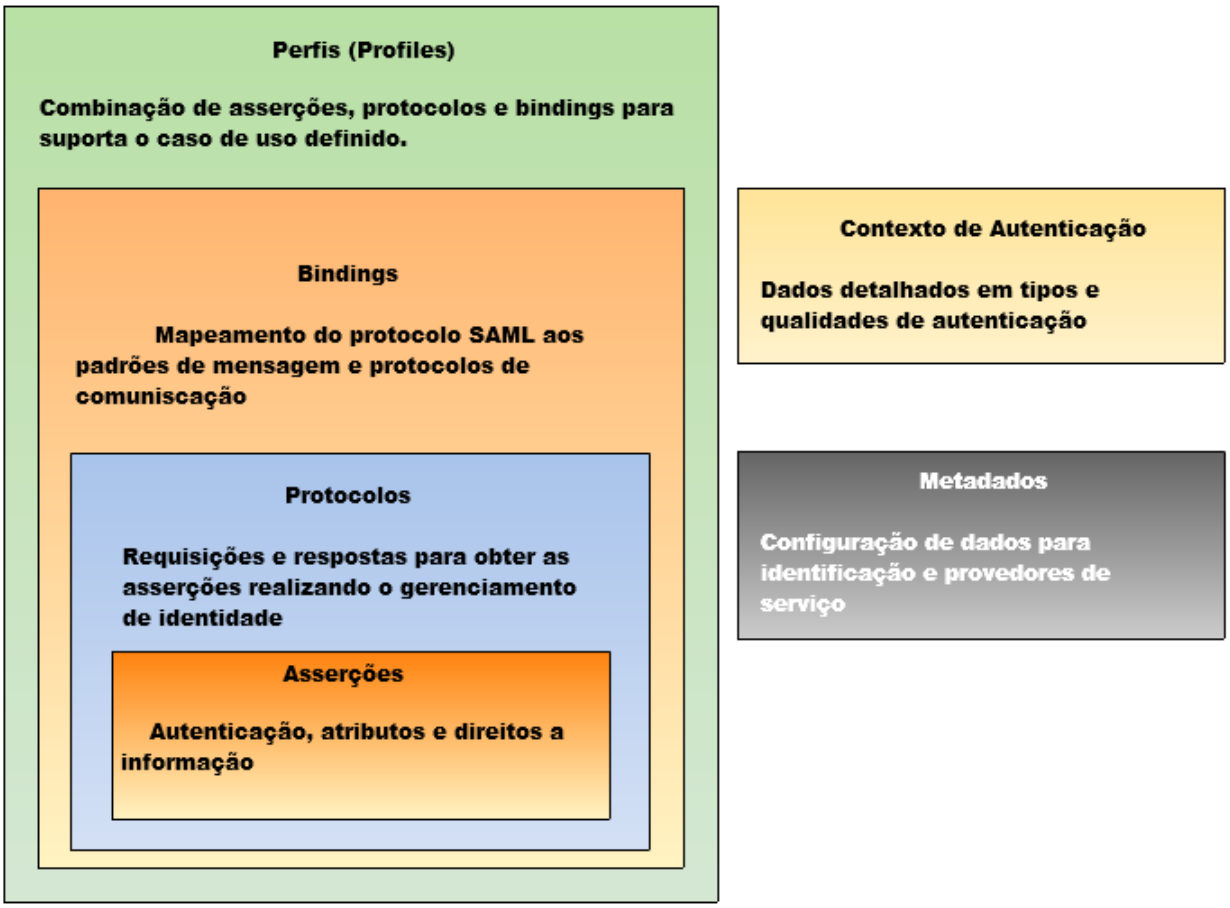

Figura 6 – Conceitos do SAML - Fonte: Ragouzis et al. (2008)

Na figura 6 é apresentada a relação entre os conceitos básicos de SAML, e um dos componentes importantes para o SAML é os Profiles. É neles em que são definidas as restrições de conteúdo para as asserções do SAML, protocolos e *Bindings* para serem resolvidos no caso de uso. Existe também atributos de Profiles que não se refere a proto-

colos de mensagem ou *Bindings*, e que define como serão realizadas a troca de informações desses atributos utilizando as asserções alinhando-se ao seu uso junto ao ambiente de uso do SAML como no caso do LDAP.

Confirmação do Sujeito

A asserção de SAML pode conter um elemento chamado ***SubjectConfirmation***. Em termos práticos, este elemento se refere ao que determinada entidade está permitida a fazer. Então a entidade está tentando usar a asserção para confirma a afirmação, de forma a implicar um relacionamento com recurso. Essa asserção pode ter inúmeros elementos de ***SubjectConfirmation***, mas a afirmação só precisa ser confirmada uma vez para todos.

O elemento ***SubjectConfirmation*** fornece a habilidade de ambas as partes verificarem a veracidade da afirmação, por exemplo um determinado serviço precisa de ter acesso a uma determinada informação que está situada em outro serviço, para isso ele envia um requisição de leitura de serviço afirmando sua necessidade, e outro serviço que disponibiliza a informação vai atestar a veracidade desta afirmação junto ao SAML e se confirmado, irá liberar para o serviço poder ler a determinada informação.

Outro elemento importante é o ***Method*** que indica o método que a parte precisa usar para verificar a afirmação, são elas:

- ***urn:oasis:name:tc:SAML:2.0:cm:holder-of-key*** – Neste modelo, recai no conhecimento de uma chave no caso publica privada que é armazenado no elemento ***SubjectConfirmationData*** usado para asserção e confirmação;
- ***urn :oasis:name:tc:SAML:2.0:cm:sender-vouches*** – Neste modelo, a parte usa um critério para determina se a parte está permitida ou não a realizar tão ação, de acordo com o resultado da asserção realizada;
- ***urn:oasis:name:tc:SAML:2.0:cm:bearer*** – Neste modelo qualquer um que for realizada a asserção estará permitido a realizar tal ação;

Componentes do SAML

O SAML ***Assertion*** permite a asserção da segurança da informação requisitada por uma parte, em forma de *Statements* (declarações) sobre o *Subject* (a outra parte). Uma asserção contém o básico requerido e informações opcionais que se aplicam a todos os *Statements* e usualmente contém a asserção do *Subject*, *Conditions* (condições) usadas para validar a asserção e as afirmações da asserção.

Existem três tipos de ***Statements*** que podem ser carregados em uma asserção, sendo o ***Authentication Statements*** para lidar com as afirmações de autenticação como a identificação do usuário e o horário da autenticação; o ***Attribute Statements***

que trata dos atributos identificadores do ***Subject***; e por fim, o ***Authorization Decision Statements*** em que se define as ações a qual o ***Subject*** identificado pode realizar.

O SAML ***Protocols*** define uma série de protocolos de requisição e resposta, e no caso do contexto de autenticação será usado o *Authentication Request Protocol* que define meios para realização da requisição de autenticação; e o *Single Logout Protocol* que define o mecanismo para sair da aplicação e fechar a sessão do usuário.

O SAML ***Bindings*** detalha como os vários protocolos de mensagem SAML poderão ser carregados pelo protocolo de transporte, e neste caso iremos usar alguns como o ***HTTP Redirect Binding***, ***HTTP POST Binding*** e o ***SAML SOAP Binding***.

O SAML ***Profiles*** define como as asserções, protocolos e *Bindings* são combinados e junto as restrições irá prover uma grande interoperabilidade em diversos cenários. Os principais perfis que iremos utilizar dentro do SAML são o ***Web Browser SSO Profile*** que define junto ao ***Authentication Request Protocol*** e o SAML, mensagens de resposta e asserções para atingir o SSO nos navegadores; e o outro perfil utilizado também será o ***Single Logout Profile***.

SAML em XML

O SAML é uma estrutura que foi escrita baseado no XML, e com isso facilita a passagem do protocolo SAML pelo protocolo de transporte, veja abaixo o exemplo figura 7 de como é a estrutura em blocos usando um protocolo de transporte junto ao SAML.

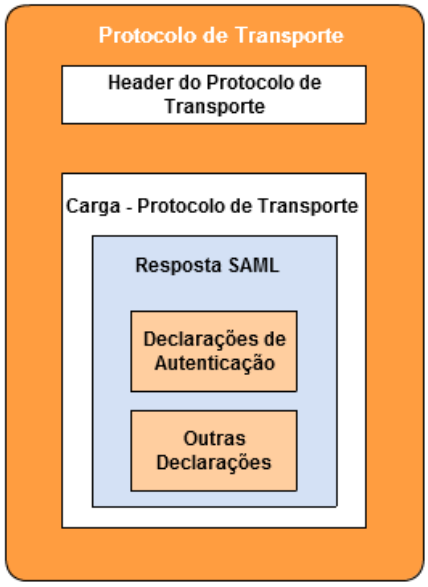

Figura 7 – Relação dos componentes SAML - Fonte: Ragouzis et al. (2008)

E na tabela 2 abaixo, é apresentado um exemplo de asserção com *Subject*, *Conditions* e *Authentication Statement*. Esta figura representa o SAML na integra em seu uso,

contendo os principais elementos expressos em XML. E na figura 8 é possível visualizar em blocos o encapsulamento do SAML no SOAP.

Tabela 2 – Exemplo de mensagem XML de asserção do SAML Fonte: Ragouzis et al. (2008)

```
1    <saml:Assertion xmlns:saml="urn:oasis:name:tc:SAML:2.0:assertion"
2        Version="2.0"
3        IssueInstant="2005-01-31T12:00:00:00Z">
4        <saml:Issuer Format="urn:oasis:names:SAML:2.0:nameid-format:entity">
5        http://idp.example.org
6        </saml:Issuer>
7        <saml:Subject>
8            <saml:NameID
9  Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10                   j.doe@example.com
11           </saml:NameID>
12       </saml:Subject>
13       <saml:Conditions
14           NotBefore="2005-01-31T12:00:00Z"
15          NotOnOrAfter="2005-01-31T12:10:00Z">
16       </saml:Conditions>
17       <saml:AuthnStatement
18           AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
19           <saml:AuthnContext
20               <saml:AuthnContextClassRef>
21                   urn:oasis:name:tc:SAML:2.0:ac:classes:PasswordProtected
22               </saml:AuthnContextClassRef>
23           </samlAuthnContext>
24       </saml:AuthnStatement>
25   </saml:Assertion>
```

- Linha 1: Começa a asserção contendo a declaração do *namespace* de asserção do SAML, que é convencionalmente representado pelo prefixo ***saml:***;
- Linha 2 a 6: Traz informações sobre a natureza da asserção, qual a versão do SAML que está sendo usada, quando a asserção foi criada e que realizou a requisição;
- Linha 7 a 12: Provê informações sobre o *Subject* da asserção. Informações como o identificador dele sob o formato de e-mail como é definido em regra;
- Linha 13 a 16: Traz as condições dessa requisição que no caso é um período de validade que o usuário requer de duração da sessão em determinado serviço;
- Linha 17 a 24: Nestas linhas é possível ver o *Authentication Statement*, apresentando que o usuário está autenticando usando um mecanismo de transporte protegido com senha e também o tempo e a data em que foi realizada a autenticação;

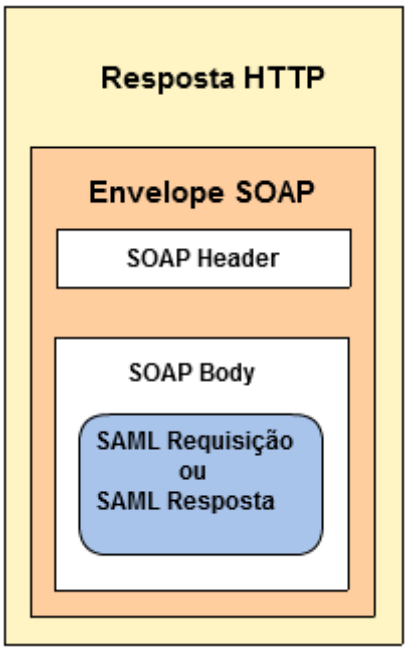

Figura 8 – Encapsulamento do SAML no SOAP via HTTP - Fonte: Ragouzis et al. (2008)

Privacidade e Segurança em SAML

Sob o contexto de tecnologia da informação, privacidade se refere a habilidade do usuário de controlar como os dados de sua identidade deverão ser compartilhados e usados, assim como mecanismos de inibição de ações em múltiplos provedores de serviços de serem inapropriadamente correlacionados. Para isso o SAML conta com mecanismos para diferentes cenários aonde os requisitos de privacidade precisam ser levados em conta. Mecanismos como: estabelecimento de pseudônimos, identificadores transientes, níveis de autorização sob determinada autenticação, consentimento de outro usuário para acessar determinado serviço, são exemplos de mecanismos que o SAML oferece para garantir a privacidade.

Em relação à segurança o SAML provê também mecanismos para atender as normas de segurança da *Security Services Technical Commitee* (SSTC) um comitê que trata de padrões de segurança em que os protocolos devem conter, e o SAML foi desenvolvido seguindo as recomenções do SSTC. Alguns dos mecanismos que o SAML conta são o uso do *Public Key Infrastructure* (PKI), apesar de não ser mandatório seu uso; realizações de requisições HTTP sob o *Secure Socket Layer* (SSL) 3.0 e o *Transport Security Layer* (TLS) 1.0; uso de certificados digitais como o X.509; e por fim, a validação de requisições por assinatura digital, são exemplos de mecanismos que garantem uma alta segurança na asserção do SAML entre as partes.

Perfil SSO para Navegadores

O perfil SSO para navegadores define a estrutura de mensagem do SAML e os *Bindings* necessários que irão ser usados. Esse perfil define uma variedade de opções, primariamente tendo duas opções:

- ***Identity Provider (IdP) Initiated:*** Quando o usuário acessa o servidor de autenticação diretamente e é redirecionado após a autenticação;
- ***Service Provider (SP) Initiated:*** Quando o usuário acessa o serviço e o serviço redireciona o usuário para o servidor de autenticação e só então após a sua autenticação ele será redirecionado ao serviço em que tentava acessa;

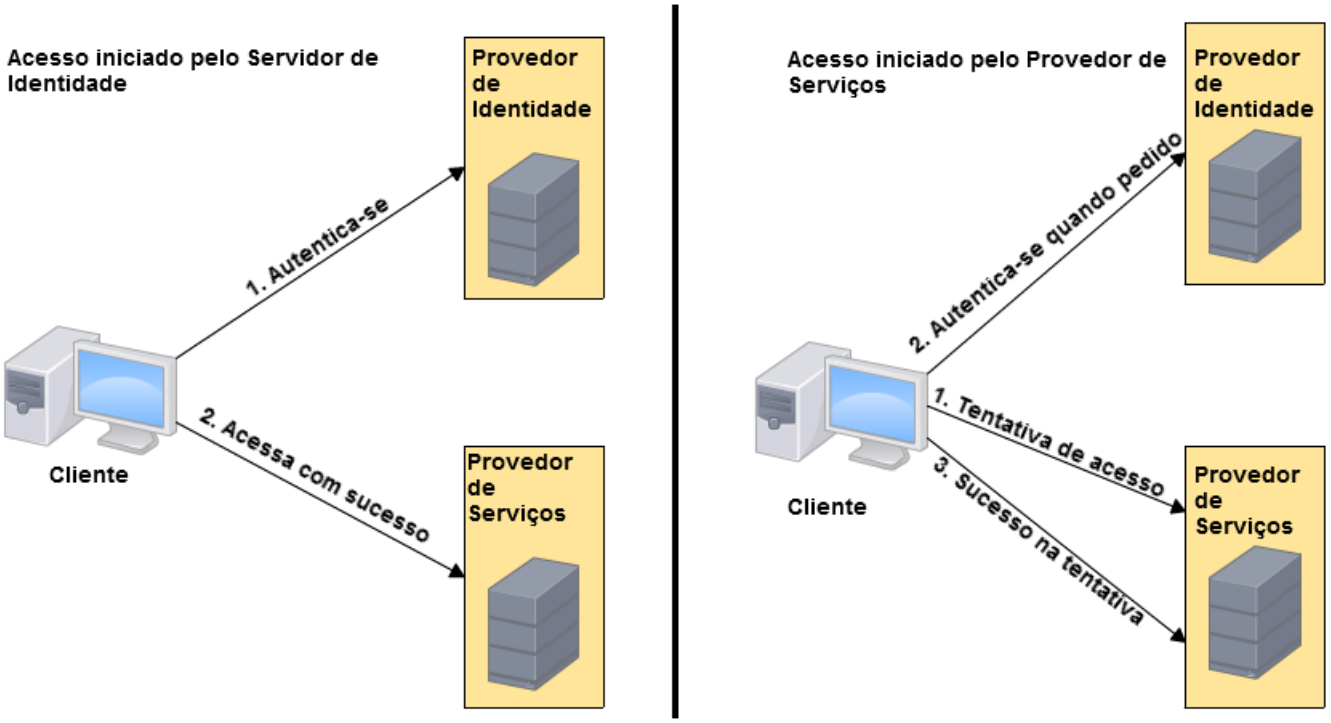

Figura 9 – Exemplo do IdP e SP - Fonte: Ragouzis et al. (2008)

De acordo com o contexto definido, o tipo de perfil SSO que será utilizado será o iniciado pelo SP, pois é o perfil que mais se encaixa de acordo com o contexto definido por Agilar (2016). Ou seja, o usuário acessa diretamente o serviço e é redirecionado para uma página geral de autenticação e só então irá saber se possui direito a acesso ou não.

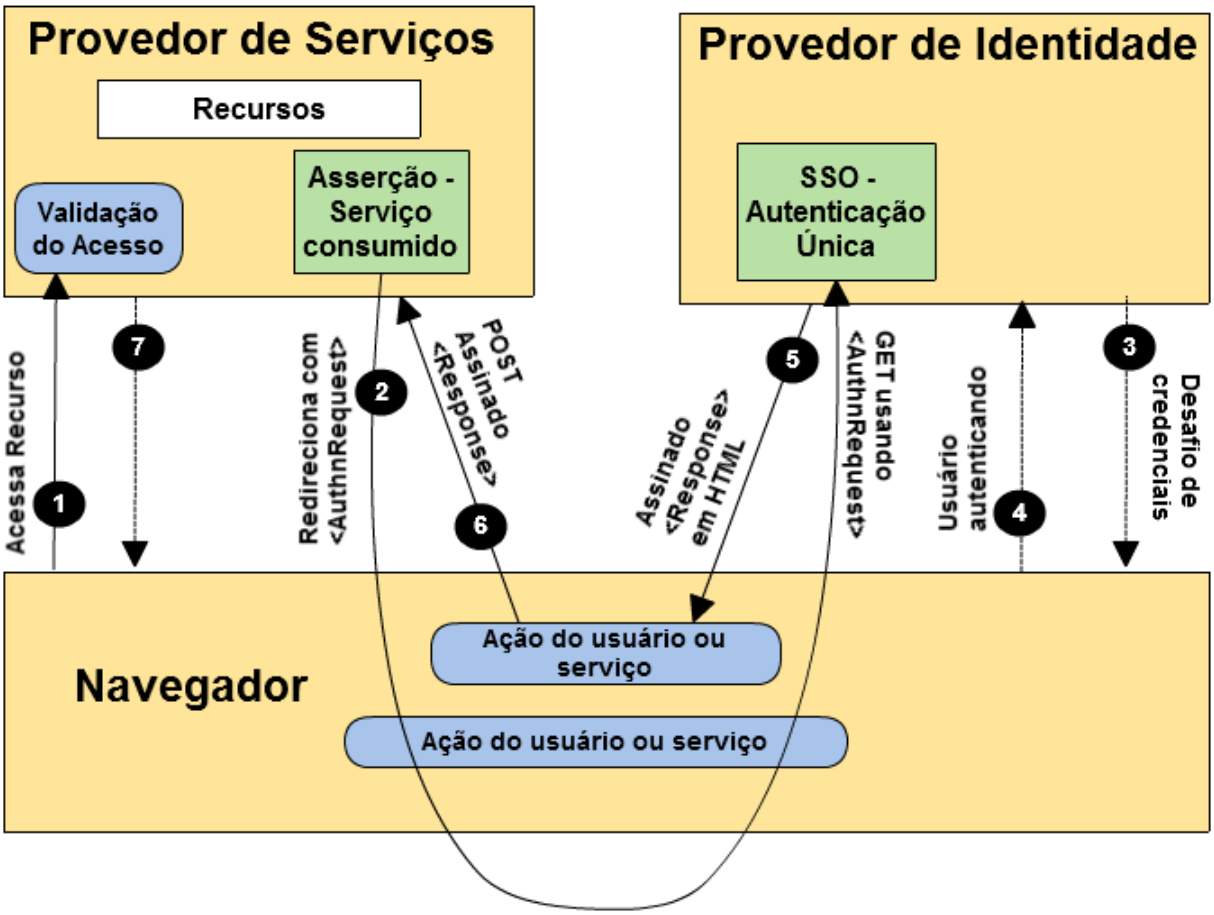

Figura 10 – Fluxo do SP Initiated - Fonte: Ragouzis et al. (2008)

Segundo Ragouzis et al. (2008), a figura 10 representa a autenticação por SP *Initiated* e ele explica essa figura da seguinte forma:

- Passo 1.: O usuário realiza uma tentativa de acessar o recurso (neste caso o exemplo está sendo referenciado como sp.example.com). O usuário não possui uma sessão de autenticação válida no site. O SP salva a requisição do recurso (no caso a URL) em um estado local em que possa ser repassado para a validação Web da autenticação única (SSO) ao longo da troca;

- Passo 2.: O SP envia uma resposta de redirecionamento ao navegador. No *Header* do redirecionamento contém a URI de destino para a autenticação única (SSO) no provedor de identidade junto com o ***<AuthnRequest>*** de forma codificada junto com a consulta da URL nomeada ***SAMLRequest***. Na tabela 3 é apresentado o XML de como é enviado a requisição do SAML;

Tabela 3 – Exemplo de XML da requisição SAML com *AuthnRequest* - Fonte: Ragouzis et al. (2008)

```
<samlp:AuthnRequest
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="identifier_1"
        Version="2.0"
        IssueInstant="2004-12-05T09:21:59Z"
        AssertionConsumerServiceIndex="1">
        <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
        <samlp:NameIDPolicy
            AllowCreate="true"
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

- Passo 3.: O serviço de SSO determina se o usuário já está usando uma autenticação no contexto de segurança no IdP que já cumpra o padrão ou os requisitos da política de autenticação. Se não, o IdP interage com o navegador realizando um desafio por credenciais para validar a integridade do usuário;

- Passo 4.: O usuário repassa as credenciais validas para autenticação no contexto de segurança criado pelo IdP para o usuário em relação ao serviço em que ele deseja acessar;

- Passo 5.: O serviço de SSO do IdP constrói a asserção do SAML representando a autenticação do usuário no contexto de segurança. Como o POST *Binding* será utilizado, a asserção precisa ser assinada digitalmente e colocada dentro da mensagem do SAML ***<Response>***. A mensagem do ***<Response>*** é colocada na forma de HTML como escondida, nomeada de ***SAMLResponse***. Para facilidade do processo, a forma HTML tipicamente será acompanhada de um script que irá realizar o POST no serviço destino com a sessão para ser validada no SP. Na tabela 5 mostra o envio da asserção de resposta do SAML ***<Response>*** pronta para ser validada pelo SP;

Tabela 4 – Exemplo de XML do *SAMLResponse* após requisição - Parte 1 - Fonte: Ragouzis et al. (2008)

```
<samlp:Response
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="identifier_2"
        InResponseTo="Identifier_1"
        Version="2.0"
        IssueInstant="2004-12-05T09:22:05Z"
        Destination="https://sp.example.com/SAML2/SSO/POST">
        <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
        <samlp:Status>
            <samlp:StatusCode
                Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
        </samlp:Status>
        <saml:Assertion
            xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="identifier_3"
            Version="2.0"
            IssueInstant="2004-12-05T09:22:06Z">
            <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
            <ds:Signature
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
            <saml:Subject>
                <saml:NameID
                    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
                    3f7b3dcf-1674-4ecd-92c8-1544f34baf8
                </saml:NameID>
                <saml:SubjectConfirmation>
                    Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                    <saml:SubjectConfirmationData
                        InResponseTo="identifier_1"
                        Recipient="https://sp.example.com/SAML2/SSO/POST"
                        NotOnOrAfter="2004-12-05T09:27:05Z"/>
                </saml:SubjectConfirmation>
            </saml:Subject>
            <saml:Conditions
                NotBefore="2004-12-05T09:17:05Z"
                NotOnOrAfter="2004-12-05T09:27:05Z"?>
              <saml:AudienceRestriction>
                    <saml:Audience>
                    https://sp.example.com/SAML2</saml:Audience>
                    </saml:AudienceRestriction>
            </saml:Conditions>
            <saml:AuthnStatement
                AuthnInstant="2004-12-05T09:22:00Z"
```

Tabela 5 – Exemplo de XML do *SAMLResponse* após requisição - Parte 2 - Fonte: Ragouzis et al. (2008)

```
                <saml:AuthnContext>
                    <saml:AuthnContextClassRef>
                        urn:oasis:names:tc:SAML:2.0:ac:classes:passwordProtected
                    </saml:AuthnContextClassRef>
                </saml:AuthnContext>
            </saml:AuthnStatement>
        </saml:Assertion>
    </samlp:Response>
```

- Passo 6.: O navegador executa o script do HTML e envia a requisição HTTP POST com a requisição de asserção para o SP consumir o *Token* da sessão e então validar ele;
- Passo 7.: Após receber o *Token* o SP valida a assinatura recebida no formato SAML e abre a sessão com o redirecionamento do navegador para a página de execução dos serviços a quais o usuário deseja;

2.4 Justificativa de escolha do SAML e SOAP

As justificativas que explicam as escolhas das tecnologias que servirão de base para o modelo de centralizado de autenticação recai no contexto que foi definido. Os sistemas da UnB são sistemas antigos e que foram sendo evoluídos ao longo do tempo até chegar em um momento em que múltiplos serviços precisavam se comunicar entre si com o objetivo de garantir a integridade dos dados de seus usuários e garantir menos erros. Muitos de seus sistemas legados foram desenvolvidos pensando no serviço como um todo, e não como uma parte (AGILAR, 2016).

Sob esse contexto foi definido o SOAP como o protocolo de comunicação entre serviços e usuários devido aos seguintes fatores: O SOAP é uma plataforma independente, ou seja, independente de linguagem, qualquer plataforma pode usar se implementado o escopo do SOAP, sem contar do fato de ser baseado em XML; a extensão de SOAP é simples, garantindo a possibilidade de implementações de segurança e de uma arquitetura de comunicação federada, sem contar de ser uma recomendação padronizada pela W3C (MUMBAIKAR; PADIYA et al., 2013). Dentre algumas vantagens do SOAP, pode se citar:

- Simplicidade;
- Portabilidade;

- Amigável ao *Firewall*;
- Usa padrões abertos e definidos por uma organização (W3C);
- Aceitação universal;
- Versátil;
- Altamente flexível;
- Apenas necessário saber o formato da requisição SOAP;
- Implementável em qualquer linguagem;

Por esses motivos e outros, dentre suas vantagens o SOAP foi levado em consideração por ser possível de se adaptar a qualquer contexto desejável com a possibilidade de no caso implementar a camada de segurança usando criptografia e dentre outros métodos de segurança (WALEED; AHMAD, 2008).

A justificativa para o SOAP também recai na utilização do SAML para autenticação. A justificativa de uso do SAML está nos seus benefícios. Ele permite sistemas de segurança e aplicações serem desenvolvidas e evoluídas independentemente. Isso porque o SAML fornece um conjunto de padrões de interface interoperáveis. Padronização de interface entre sistemas permite uma integração mais confiável, barata e rápida. Com mais perfil de SAML sendo desenvolvidos, esses benefícios serão abertos a diferentes tipos de gerenciamento ao acesso (GROSS, 2003).

Desenvolvedores de sistemas seguros se beneficiam de existir padrões no esquemas e protocolos para expressarem a segurança da informação. Assim como o baixo acoplamento do software com a camada de segurança da infraestrutura, os usuários finais serão beneficiados pelo SSO do SAML com uma experiência personalizada e com uma privacidade mais amigável e confiável (RAGOUZIS et al., 2008). Dentre esses fatores, as principais vantagens que levaram a escolha do SAML para este trabalho foi:

- **Plataforma neutra:** O SAML abstrai o arcabouço da segurança para longe da arquitetura da plataforma. Tornando a segurança mais independente da lógica da aplicação, como propõe o princípio da arquitetura SOA, segurança como um serviço;
- **Melhoria na experiência online do usuário final:** O SAML SSO permite ao usuário autenticar em um IdP e então acessar o SP sem a necessidade de uma autenticação adicional. Em adição, a identidade federada (compartilhamento de informações entre aplicações) com o SAML permite uma melhor customização para a experiência do usuário promovendo uma privacidade única a cada serviço;

- **Baixo acoplamento entre os diretórios:** O SAML não precisa manter a informação do usuário nem sincronizada entre os diretórios;
- **Custo administrativo reduzidos para os SP's:** Usando o SAML para o reuso do ato de autenticação reduz o custo de manter informação do usuário em múltiplas aplicações, sem contar de torna melhor o gerenciamento e a integridade das informações, transferindo todo esse peso para um servidor exclusivo de identidade;
- **Tranferência de riscos:** O SAML pode agir para empurrar a responsabilidade para um melhor gerenciamento de identidades para o provedor de identidades, o que é mais compatível com um modelo de negócios do que um modelo orientado um único provedor de serviços;

3 Desenvolvimento

Este Capítulo irá apresentar a modelagem centralizada de autenticação como serviço em um barramento, bem como a implementação de pseudo serviços que simularam um ambiente de produção com múltiplos usuários e utilizando uma solução de IdP para gerenciar a autenticação nesses serviços. A seção 3.1 irá tratar de explicar a modelagem da solução centralizada de autenticação. Na seção 3.2 é explicado o fluxo de dados da troca de mensagens do SAML entre SP, usuário e IdP. Na seção 3.3 é apresentado a organização do trabalho, ou seja, as tecnologias utilizadas para o desenvolvimento dos pseudo serviços. A seção 3.2 contém um subtópico para tratar da explicação sobre a implementação da solução. Na seção 3.4 é explicado a montagem do ambiente de testes, e na seção 3.5 a definição dos critérios de segurança bem como os cenários dos testes a serem executados. Por fim, na seção 3.6 é realizado a avaliação dos resultados obtidos.

3.1 Solução para Modelagem Centralizada de Autenticação

A solução para a modelagem centralizada de autenticação segue uma série de critérios para centralizar a utilização da autenticação em múltiplos serviços. O primeiro critério é a utilização de um servidor de autenticação que irá centralizar todas as autenticações de todos os serviços; em seguida, o compartilhamento de sessões após a autenticação será de grande importância no caso de acessar múltiplos serviços sem a necessidade de se identificar novamente, e por fim uma base de dados centralizada, compartilhada entre os múltiplos serviços correlacionada somente a identificação de perfis e usuários e seus atributos para realizar as devidas autorizações.

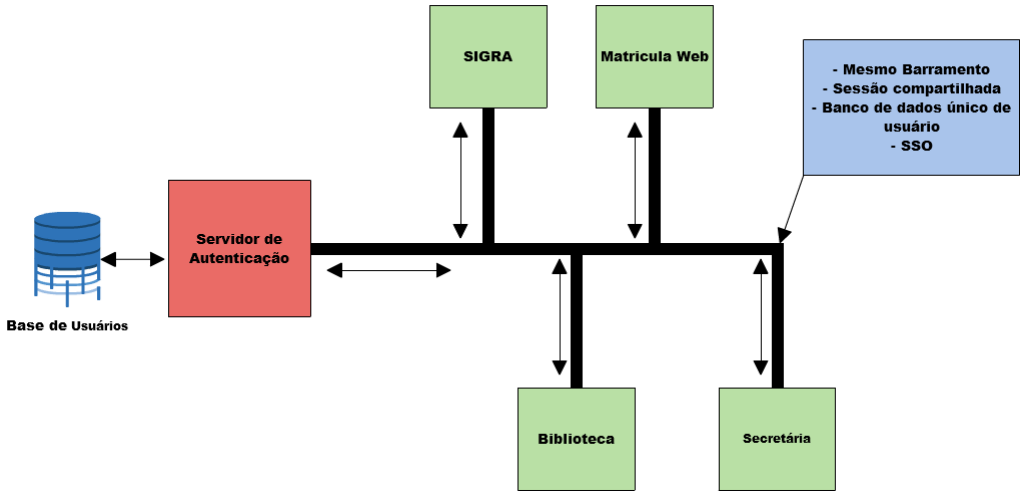

Figura 11 – Modelagem Centralizada de Autenticação - Fonte: Autor

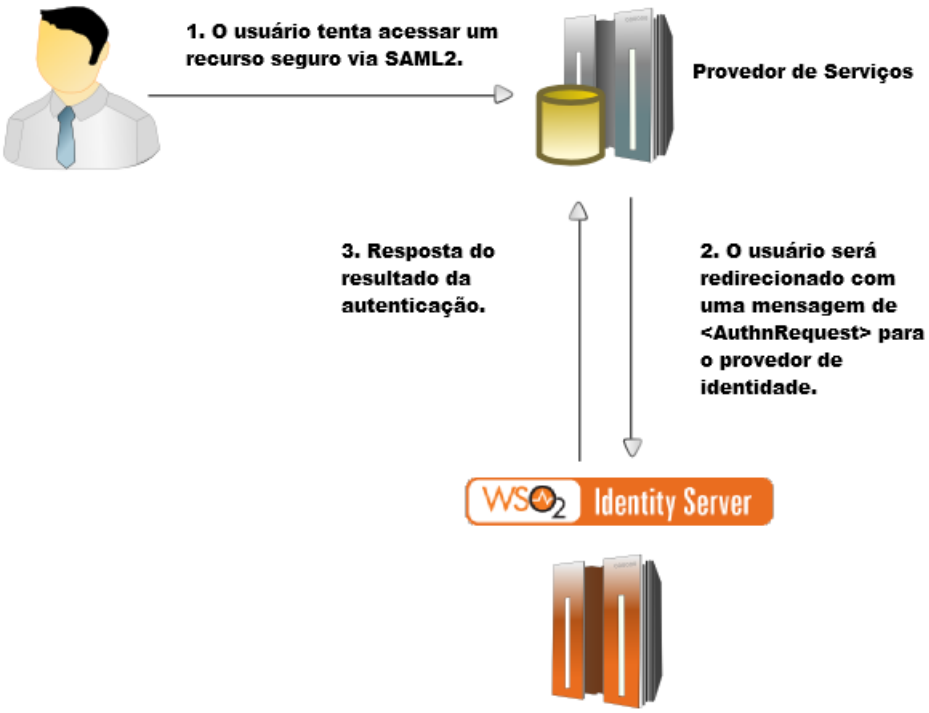

Figura 12 – Fluxo de autenticação - Fonte: Attanayake (2010)

O desenvolvimento da solução proposta é dividida em dois pontos: Dois pseudo provedores de serviços para simular os serviços da UnB e uma solução já implementada utilizando o WSO2 *Identity Server*. O WSO2 é um provedor de serviços completo que contempla toda a parte de autenticação, pronto para agir como um *Central Authentication Service* (CAS), ou seja, um serviço independente de autenticação que pode se conectar a uma base de dados já existente ou utilizar a própria base. Ele permite gerenciar múltiplas base de dados ou utilizar uma base centralizada, além de interagir com múltiplos IdP's se necessário. É possível ver um pequeno diagrama generalizado de como ficará a implementação geral na figura 11 e com a utilização do WSO2 na figura 12.

3.2 Fluxo de Dados da Troca de Mensagem

O principal núcleo deste trabalho está na trocas e asserções das mensagens com o objetivo de autenticar o usuário no provedor de identidade para ter acesso ao serviço e ou recurso. A figura abaixo representa o principal fluxo de autenticação que é seguido para autenticar todos os usuários para que possam ter acesso aos serviços que o barramento oferece, conforme pode ser visto no diagrama de sequência na figura 13. E logo em seguida será apresentado os modelos de mensagem do *<AuthnRequest>*, *SAMLResponse* e do *Signed SAMLResponse*.

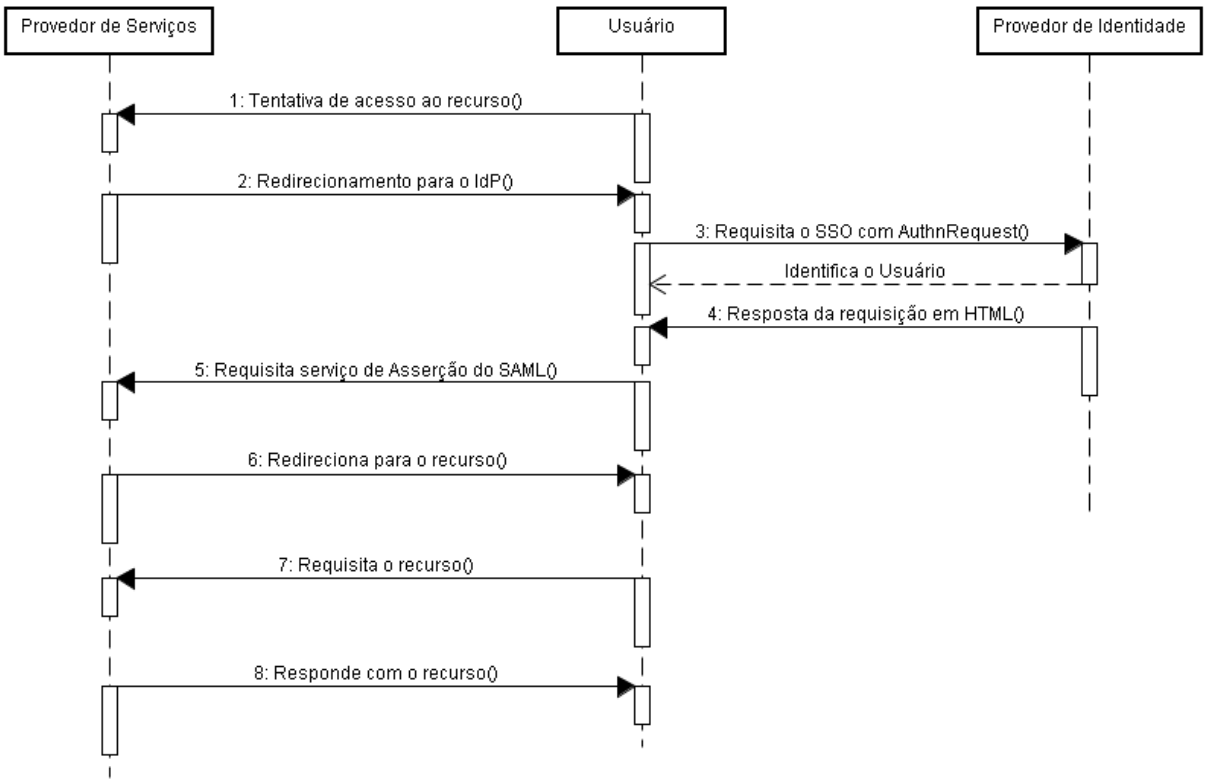

Figura 13 – Diagrama de Sequência do SAML HTTP-POST - Fonte: Gajasinghe (2017)

Indo passo a passo no diagrama de sequência na figura 13:

1. **Tentativa de acesso ao recurso** – O usuário realiza uma tentativa de acesso ao recurso do serviço, o serviço verifica que o usuário não possui uma sessão válida;

2. **Redirecionamento para o IdP** – O provedor de serviço cria então uma mensagem no formato especificado na tabela 18, e repassa ao usuário com HTTP-POST atrelado a um redirecionamento para o servidor de identidade;

3. **Requisita o SSO com *AuthnRequest*** – O usuário é redirecionado ao servidor de identidade com uma mensagem requisitando o acesso ao recurso;

    - **Identifica o Usuário** – O usuário entra com suas credenciais no servidor, que o identifica em sua base de dados;

4. **Resposta da Requisição em HTML** – O servidor de identidade entrega um *Token* para o usuário em forma de XML representado nas tabelas 19 e 21 que é retratado novamente em HTML conforme a tabela 18 mostra, contendo o *SAMLResponse*;

5. **Requisita serviço de Asserção do SAML** – O navegador do usuário é redirecionado a uma página conforme mostrado na tabela 18 via HTTP-POST com o *SAMLResponse* para ser validado;

6. **Redireciona para o Recurso** – Após a asserção, é validado a assinatura e o serviço libera a sessão ao usuário;

7. **Requisição do recurso** – O usuário então realiza a requisição formal do recurso novamente;

8. **Responde com o recurso** – Com a sessão valida o provedor de serviço entrega o recurso ao usuário;

Com base na explicação do diagrama de sequência, as mensagens trocadas entre SP e IdP utilizam uma *Public Key Infrastructure* (PKI). O PKI consiste em uma série de padrões, protocolos e serviços que permitem ao usuário autenticar um ao outro usando certificados digitais emitidos por um *Certification Authority* (DFT, 2015). O padrão de PKI que o SAML utiliza é os certificados X509, que utilizam para validar tanto a aplicação quanto o usuário na troca de mensagens. É possível visualizar o certificado bem como a digestão de parte da mensagem e assinatura nas tabelas 19 e 21 referentes ao *SAMLResponse*.

3.3 Organização do Projeto

A implementação dos pseudo serviços foi realizada utilizando com base na documentação do WSO2 [1] para utilizar seu IdP, e implementar a parte do SP para utilizar o IdP da forma correta. A tecnologia usada para desenvolver os pseudo serviços foram os *Servlets*, classes Java que tratam as requisições Web. Juntamente com o Maven, um gerenciador de dependências, e algumas API's que são sugeridas pelo WSO2 e a principal delas, a API de integração do IdP com o SP fornecida pela WSO2. Também é usado as *Java Server Pages* (JSP) com alguns *Scriptlets* (Código Java embutido na JSP), JSP's são páginas dinâmicas que criam documentos HTML e ou XML para manipulação de forma dinâmica. O framework utilizado para o desenvolvimento foi o *Integrated Development Environment* (IDE) Eclipse, que é a IDE mais usada pelos desenvolvedores Java. Em resumo, das tecnologias utilizadas:

- **Java JDK/JRE 1.8**;
- **Eclipse Neon**;
- **Java *Servlets***;

[1] https://docs.wso2.com/display/IS530/WSO2+Identity+Server+Documentation

- **Maven**;
- **Java *Server Pages***;
- ***Scriptlets***;
- **WSO2 *Identity Server* - API**;
- **Apache Tomcat 9**;

3.3.1 Implementação da Solução

Para o desenvolvimento deste projeto está sendo utilizado um layout de projeto chamado *Dynamic Web Project* (DWP) do Eclipse, que organiza todo o sistema de pastas e código-fonte em uma padronização utilizada pelos desenvolvedores. O projeto usa JSP versão 2.5 por questões didáticas para melhor entendimento e facilidade no desenvolvimento, porém não é recomendado a utilização desta versão em produção em função de estar desatualizada. A organização do projeto pode ser visto na figura 14.

O tratamento das URL's nas *Servlets* é realizado conforme a configuração do *web.xml* e o tratamento do *Assertion Consumer Service* do SAML é feito na JSP, para averiguar se a assinatura, tanto do usuário quanto da aplicação estão de acordo com o certificado do IdP, e caso estejam essa é a condição para liberar a sessão do usuário para acessar o recurso.

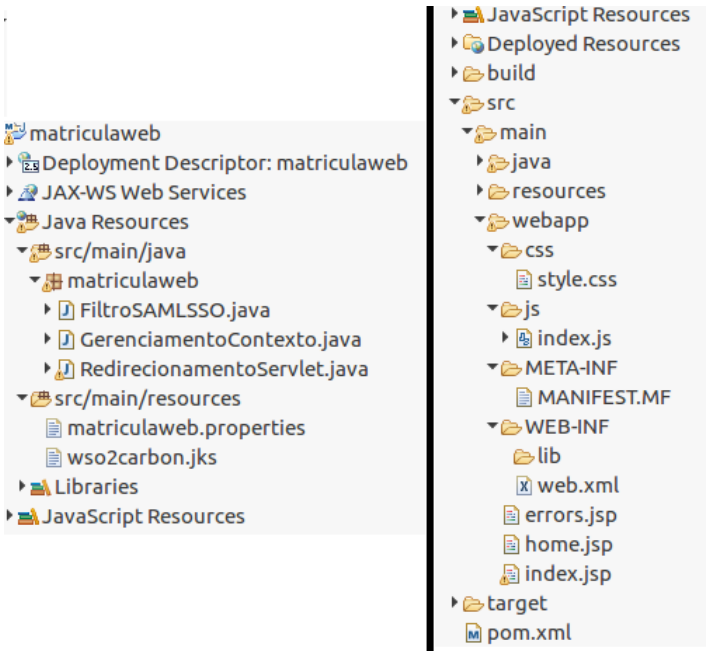

Figura 14 – Organização do projeto - Fonte: Autor

O projeto é dividido em 3 classes Java para o tratamento das requisições Web via Servlet; conta com 2 recursos para configurações e por fim 3 páginas JSP para simulação

de entrada em um determinado serviço. Esse mesmo código foi replicado com mudança de visual para o serviço de simulação SIGRA. Segue a explicação de cada item de acordo com as figuras acima:

- **GerencimanetoContexto.java -** Responsável por inicializar o contexto do serviço, com a leitura das propriedades do SAML no matriculaweb.properties e também por carregar o certicado X509 usado pelo servidor de identidade;

  Esta classe é principal por realizar as instanciações dos objetos e seus atributos que serão colocados em formato de mensagem no SAML para realização das requisições no formato de *Authn*, é possível ver esse código no seguinte trecho da classe na tabela 6.

Tabela 6 – Método *contextInitialized* - Fonte: Autor

```
public void contextInitialized(ServletContextEvent servletContextEvent) {
        try {
            if(servletContextEvent.getServletContext()
            .getContextPath().contains(PATH)) {
                propriedades.load(servletContextEvent.getServletContext()
                .getResourceAsStream(PATH_PROPERTIES));
            }
            SSOAgentConfig config = new SSOAgentConfig();
            config.initConfig(propriedades);
            config.getSAML2().setSSOAgentX509Credential(
            associandoCredencial(servletContextEvent.getServletContext().
                    getResourceAsStream(PATH_CERT), propriedades));
            servletContextEvent.getServletContext().setAttribute(SSOAgentConstants.
            CONFIG_BEAN_NAME, config);
        } catch (IOException e){
            logger.log(Level.SEVERE, e.getMessage(), e);
        } catch (SSOAgentException e) {
            logger.log(Level.SEVERE, e.getMessage(), e);
        }
    }
```

  Neste método está sendo definido o contexto da servlet com um objeto SAML de configuração que contém as propriedades de configuração e o certificado para serem enviados ao IdP.

- **FiltroSAMLSSO.java -** A cada requisição no serviço é realizado um tratamento e uma filtragem da requisição para adicionar e ou carregar dados referentes ao protocolo de autenticação SAML;

  No código da tabela 7 temos um trecho da classe, a qual realiza o tratamento das requisições. Este método realiza a configuração do HTML para recebimento

Tabela 7 – Método *doFilter* - Fonte: Autor

```
public void doFilter(ServletRequest servletRequest, ServletResponse
    servletResponse,
                   FilterChain filterChain) throws IOException,
                       ServletException {

    String binding =
        servletRequest.getParameter(SSOAgentConstants.SSOAgentConfig.
    SAML2.HTTP_BINDING);
    if(binding == null || binding.isEmpty() ||
        !("HTTP-POST".equals(binding))){
     binding = HTTP_POST_BIND;
    }
    SSOAgentConfig config =
        (SSOAgentConfig)filtroConfig.getServletContext().
    getAttribute(SSOAgentConstants.CONFIG_BEAN_NAME);
    config.getSAML2().setHttpBinding(binding);
    redirecionamento(config, servletRequest);
    servletRequest.setAttribute(SSOAgentConstants.CONFIG_BEAN_NAME,config);
    super.doFilter(servletRequest, servletResponse, filterChain);
}

public String postHtml(String usuario, String senha, String samlidpurl)
    throws UnsupportedEncodingException {
  String autorizacao = usuario + ":" + senha;
  autorizacao = new String(Base64.encode(autorizacao.getBytes(ENCODE)));
  return HTML_A + samlidpurl + HTML_B + autorizacao + HTML_C;
}
```

do *SAMLResponse* responsável por carregar o *token* que irá gerar a sessão para o usuário;

- **RedirecionamentoServlet.java -** Classe Java responsável pelo redirecionamento correto dos recursos as devidas páginas jsp;

Tabela 8 – Método *service* - Fonte: Autor

```
 protected void service(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    if(request.getRequestURI().endsWith(SAML_URI)){
        request.getRequestDispatcher(HOME_URI).forward(request,response);
    } else if (request.getRequestURI().endsWith(LOGOUT_URI)){
        request.getRequestDispatcher(INDEX_URI).forward(request,response);
    }
}
```

O método *service* realiza o tratamento da *servlet*, neste caso de redirecionamento

de páginas para seus respectivos jsp conforme o for redirecionado tanto para a URL do IdP quanto do SP.

- **matriculaweb.properties -** Neste arquivo são definidas as propriedades do SAML que são carregadas por uma classe Java, que monta o objeto e instância para montagem da mensagem de requisição;
- **wso2carbon.jks -** Certificado guardado em *Java Key Store* (JKS) para identificar que este serviço é confiável no servidor de identidade, essas informações são carregadas e repassadas no escopo da mensagem para que o IdP possa identificar a aplicação e também se a requisição é confiável;
- **css/style.css -** Arquivo de tratamento visual das páginas;
- **web.xml -** Este arquivo representa o mapeamento das *Servlets* e dos *Filters*, além do mapeamento é configurado os *Listeners* que vão realizar a inicialização dos contextos necessários da aplicação;
- **errors.jsp -** Página de erro;
- **home.jsp -** Esta página é a página pós login e possui um pequeno mecanismo para averiguar se a sessão é válida e se o *SAMLResponse* corresponde ao usuário correto, tratado via *Scriptlets*;

  Neste *scriptlet* definido no JSP, é realizada a asserção da sessão do usuário. Entregue pelo IdP via classe Java FiltroSAMLSSO, a JSP contém uma sessão com os objetos do IdP que vieram via HTTP-POST, a qual irá permitir ao *scriptlet* que leia os atributos recebidos para saber se a autenticação ocorreu com sucesso, e caso tenha ocorrido, será liberado a sessão com o recurso ao usuário requisitante.
- **index.jsp -** Página de login inicial da aplicação da onde irá partir a requisição do AuthnRequest do usuário para o IdP;
- **pom.xml -** Arquivo de configuração das dependências do projeto web via Maven;

3.4 Ambiente de Teste

Para a realização de testes foi implementado dois pseudo serviços: SIGRA e Matricula Web para fins de teste do SSO. Em seguida foi feito o *download* do WSO2 *Identity Server*[2] que é usado como provedor de identidade neste caso. Após o *deploy* do WSO2 *Identity Server* foi necessário realizar a configuração das propriedades tanto nos dois projetos quanto, realizar o cadastro das aplicações no IdP. Para isso foi realizado os seguintes passos:

[2] http://wso2.com/identity-and-access-management

Tabela 9 – Serviço básico de asserção da sessão via *Scriptlets* - Fonte: Autor

```
<%String identificador = null;%>
<%if(request.getSession(false) != null &&
   request.getSession(false).getAttribute(SSOAgentConstants.SESSION_BEAN_NAME)
       == null){%>
      <%request.getSession().invalidate();%>
      <script type="text/javascript">
      location.href = "index.jsp";
      </script>
<%return;}%>

<%SSOAgentConfig ssoAgentConfig = (SSOAgentConfig)getServletContext().
getAttribute(SSOAgentConstants.CONFIG_BEAN_NAME);%>
<%LoggedInSessionBean sessaoBean =
    (LoggedInSessionBean)session.getAttribute(SSOAgentConstants.SESSION_BEAN_NAME);%>
<%LoggedInSessionBean.AccessTokenResponseBean accessTokenResponseBean =
    null;%>

<%if(sessaoBean != null){%>
   <%if(sessaoBean.getSAML2SSO() != null) {%>
      <%identificador = sessaoBean.getSAML2SSO().getSubjectId();%>
      <%accessTokenResponseBean =
          sessaoBean.getSAML2SSO().getAccessTokenResponseBean();%>
   <%} else {%>
         <script type="text/javascript">
            location.href = "index.jsp";
         </script>
         <%return;}%>
   <%} else {%>
      <script type="text/javascript">
         location.href = "index.jsp";
      </script>
  <%return;}%>
```

1. Iniciar o servidor de identidade WSO2;
2. Logar na página https://localhost:9443/carbon com usuário e senha admin;
3. Na seção de *Service Provider*, clicar em *Add*;
4. Informar o nome do SP;
5. Clicar em *Inbound Authentication Configuration* e depois em *SAML2 Web SSO Configuration* e depois em *Add*;
6. Adicionar de quem vem a solicitação, o *Issuer*;
7. Adicionar a URL que possui o serviço de Asserção Consumida, o *Assertion Consurmer URLs*;

8. Ativar *Response Signing* pois está sendo usado um certificado;
9. Ativar o *Single Logout*;
10. Salvar e dar update na configuração e repetir as mesmas opções para o outro serviço;

Na figura 15 é possível visualizar os passos 2, 3, 5, 6, 7, 8 e 9 de forma resumida, a qual retrata a configuração do SP no IdP maneira bem simples e fácil. Após levantar e configurar os SP's no IdP, é necessário dar o *deploy* nas duas aplicações, para isso foi necessário configurar o Tomcat 9 no Eclipse, adicionando as duas aplicações para executar no Tomcat, e em seguida subir as aplicações para a realização dos testes a seguir.

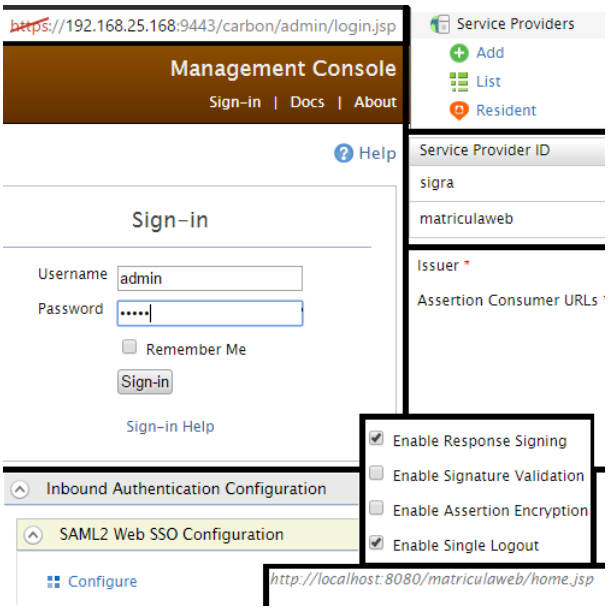

Figura 15 – Imagens relacionada a configuração do SP no IdP Fonte: Autor

3.5 Testes da Modelagem Centrelizada

Quando se fala em realizar teste de software, pode-se inferir que o objetivo é verificar se o software está de acordo com os seus requisitos. Teste também é considerado o ato de executar um software com a intenção de encontrar bugs, bem como determinar se o sistema ou programa alcança o resultados desejados a partir de sua avaliação (RIOS; MOREIRA, 2006). De acordo com Rios e Moreira (2006), teste de software é o processo que visa a execução controlada do software, com o objetivo de avaliar o comportamento no que foi designado. A execução dos testes é considerada uma forma de validar o que foi proposto.

Este trabalho procura responder a seguinte pergunta: A Modelagem Centralizada de Autenticação atende as necessidades de segurança na autenticação em um barramento de serviços? Ela oferece segurança na integridade da identidade do usuário e ou serviço

após sua autenticação? E para responder essas perguntas foram definidos dois tipos de teste: Aceitação e Segurança, que serão utilizados para validar este trabalho.

3.5.1 Teste de Aceitação

O teste de aceitação é um tipo de teste cujo objetivo é verificar se a solução atende as regras negócio previamente definidas, com base nos requisitos do software a qual está sendo testado. O teste de aceitação engloba testes funcionais, em que seu objetivo é realizar testes caixa preta, ou seja, dada uma entrada, espera-se uma determinada saída (MALDONADO et al., 2007).

Tabela 10 – Testes de Aceitação para validação - Fonte: Autor

ID	**Descrição**	**Pré-Requisitos**	**Roteiro**	**Resultados Esperados**
TA01	Realizar autenticação e deslogar da sessão.	1 - Levantar WSO2 IdP; 2 - Levantar SP;	1 - O usuário deve tentar acessar o recurso; 2 - O usuário entrará com as credenciais para autenticação; 3 - O usuário irá se deslogar;	Autenticação e deslogar da sessão com sucesso.
TA02	Realizar a autenticação em um serviço e depois tentar acessar outro serviço	1 - Levantar WSO2 IdP; 2 - Levantar SP;	1 - O usuário deve tentar acessar o recurso do serviço SIGRA; 2 - O usuário deve entrar com suas credenciais para logar; 3 - O usuário irá acessar o recurso MatriculaWeb; 4 - O usuário irá se deslogar usando o recurso matriculaweb;	O usuário após se autenticar deve ter acesso aos dois serviços sem a necessidade de se autenticar novamente;

Foram elaborados dois testes de aceitação, conforme pode ser visto na tabela 10. Esses testes visam tratar da funcionalidade do protocolo de modo geral, a funcionalidade de SSO SAML como um todo com a utilização de dois serviços como SP's, um IdP WSO2 e um usuário tentando acessar os recursos.

3.5.2 Teste de Segurança

O teste de segurança tem como objetivo validar a capacidade de proteção do software contra o acesso interno ou externo de forma não autorizada (RIOS; MOREIRA, 2006), mas sob o contexto definido, o objetivo é avaliar a segurança do SAML.

Segundo a Oracle (2010), a segurança de uma aplicação deve ser contemplada sob os seguintes eixos: confidencialidade, integridade, autenticidade, estrutura de mensagem e gerenciamento de confiança. Mas não somente isso, também é necessário garantir uma chave segura para troca entre cliente e servidor. Dentre os itens que a Oracle (2010) destaca estão:

- Troca de mensagens em um meio seguro;
- Suporte a certificado digital para assinar as requisições das aplicações;
- Sessão compartilhada entre os múltiplos serviços;
- Proteção a roubo de identidade para acesso de serviço não autorizado;

Tabela 11 – Testes de Segurança para validação - Fonte: Autor

ID	**Descrição**	**Pré-Requisitos**	**Roteiro**	**Resultados Esperados**
TS01	O sistema de WSO2 IdP deve oferecer proteção contra ataques de roubo de identidade por exemplo: *Spoof Attacks*.	1 - Levantar WSO2 IdP; 2 - Alterar no *properties* do serviço o Id-PURL ao invés de localhost para o ip da maquina; 3 - Levantar SP;	1 - Acessar serviço; 2 - Entrar com as credenciais para se autenticar;	O DEBUG do WSO2 deve identificar a possibilidade de um *Spoof Attack* e não deve permitir a autenticação do usuário ao serviço.
TS02	O sistema de WSO2 deve utilizar X509 para assinar e garantir a integridade do serviço e do usuário que está tentando autenticar	1 - Levantar WSO2 IdP; 2 - Levantar SP; 3 - Habilitar o modo *DEBUG* do IdP;	1 - O usuário deve acessar ao recurso; 2 - O usuario deve se autenticar com suas credenciais;	É esperado que o *DEBUG* mostre a assinatura e o X509 utilizado.
TS03	O sistema de WSO2 deve utilizar uma função *Hash* para digerir a assinatura e o X509.	1 - Levantar WSO2 IdP; 2 - Levantar SP; 3 - Habilitar o modo *DEBUG* do IdP;	1 - O usuário irá tentar acessar o recurso; 2 - O usuário deverá se autenticar com suas credenciais;	O *DEBUG* deve mostrar as transformações que o *SAMLResponse* utilizou para digerir a assinatura e o certificado.

Os testes de segurança elaborados na tabela 11, são condizentes com com os itens tratados pela Oracle (2010) para garantir segurança, integridade e confiança no processo de autenticação centralizado.

3.6 Resultados e Análise

O ambiente de teste foi colocado em ação para realização dos testes definidos. Os testes que foram definidos no escopo, procuram atender as necessidades justificadas no contexto deste trabalho de garantir um mecanismo funcional e seguro de autenticação para ser implantado em um barramento SOA. Os resultados estão disponíveis na tabela 12, bem como resultado obtido por cada teste será apresentado.

Tabela 12 – Resultado dos Testes - Fonte: Autor

ID do Teste	**Sucesso/Falha**
TA01	Sucesso
TA02	Sucesso
TS01	Sucesso
TS02	Sucesso
TS03	Sucesso

- **TA01 -** O objetivo deste teste é garantir a funcionalidade mais básica de todo o sistema que é a de autenticação, para identificar o usuário no sistema em que o recurso está sendo acessado. Para isso o usuário **pauloesb** foi criado para acessar as aplicações. Ao tentar acessar o recurso na URL *http://localhost:8080/sigra* o usuário foi redirecionado para *http://localhost:8080/sigra/home.jsp* que contém o serviço de *Assertion Consumer* do SAML para validar a autenticação do usuário;

  Averiguando que o usuário não estava autenticado, conforme pode ser visto na tabela 13, o mesmo foi redirecionado para o *http://localhost:8080/sigra/index.jsp*, conforme é apresentado na figura 16 aonde teve de inserir as credenciais para autenticação, logo em seguida, foi redirecionado novamente para o *https://localhost:9443/auth* apresentado na figura 17, utilizando o *AuthnRequest*. Após a autenticação com a validação do *AuthnRequest* o usuário é redirecionado para a página de pós login e só então executa a ação de deslogar, ou seja o resultado do teste TA01 é sucesso.

Tabela 13 – Log do WSO2 após tentativa de acesso a um recurso sem autenticar - Fonte: Autor

```
[2017-04-21 23:21:17,145] DEBUG
   {org.wso2.carbon.identity.sso.saml.util.SAMLSSOUtil} -
Request message <?xml version="1.0" encoding="UTF-8"?><samlp:AuthnRequest
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   AssertionConsumerServiceURL="http://localhost:8080/sigra/home.jsp"
Destination="https://localhost:9443/samlsso" ForceAuthn="false"
   ID="ihhihfeefbeikdfpnhgnnfjolhdmjfelkigdlnad"
IsPassive="true" IssueInstant="2017-04-22T02:21:16.020Z"
   ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Version="2.0"><samlp:Issuer
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion">sigra</samlp:Issuer
saml2p:NameIDPolicy xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
   AllowCreate="true"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
   SPNameQualifier="Issuer"/><saml2p:RequestedAuthnContext
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
   Comparison="exact"><saml:AuthnContextClassRef
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">urn:oasis:names:tc:SAML:2.0:ac
</saml:AuthnContextClassRef></saml2p:RequestedAuthnContext></samlp:AuthnRequest>
[2017-04-21 23:21:17,791] DEBUG
   {org.wso2.carbon.identity.sso.saml.validators.SPInitSSOAuthnRequestValidator}
   - Authentication Request Validation is successful..
[2017-04-21 23:21:17,939] DEBUG
   {org.wso2.carbon.identity.sso.saml.servlet.SAMLSSOProviderServlet} -
   Unauthenticated User
[2017-04-21 23:21:18,066] DEBUG
   {org.wso2.carbon.identity.sso.saml.servlet.SAMLSSOProviderServlet} -
   samlsso_response.html
```

- **TA02 -** Esse teste tem como objetivo garantir a principal funcionalidade e o que foi proposto ao modelo centralizado. A autenticação única (SSO), com utilização de compartilhamento de sessões entre serviços para que o usuário não necessite ficar autenticando toda vez que acessar diferentes serviços no mesmo barramento. O usuário se autenticou no serviço SIGRA, o usuário recebe o *token* via HTML com um *script* de redirecionamento para o serviço, conforme pode ser visualizado no log 14;

  O redirecionamento para o home.jsp aciona o *Assertion Consumer Service* para averiguar a validade do *token*, e só então libera a sessão ao usuário para consumir o recurso, após a validação é possível ver a página do recurso liberada ao usuário na figura 18. Depois disso o usuário acesso o outro serviço no caso do matrícula Web, com a mesma seção e obteve sucesso pois a sessão já estava liberada devido ao *token* estar ativo, sem contar que ambas aplicações utilizam o mesmo JKS permitindo o

compartilhamento da mesma assinatura para criação de uma sessão, ou seja garantindo o SSO, conforme visto na figura 19, o usuário obtém sucesso no acesso ao serviço do matrícula web, logo validando com sucesso este teste.

- **TS01 -** Este teste visa testar a reação do sistema no caso de um *Spoof Attack*, ou seja, um serviço ou usuário tentando se mascarar por outro para tentar obter acesso de autenticação do sistema, quando o indivíduo não tem acesso. O cenário foi realizado mudando a URL de asserção do matrícula web ao invés do *localhost*, para o IP do IdP e os resultados foram positivos, pois o IdP identificou uma possibilidade de *Spoof Attack* e não liberou a assinatura do *SAMLResponse*, resultado que pode ser visto na tabela 15;

  O aplicativo realiza uma asserção novamente com base na configuração definida, e pode se ver que o IdP identifica que a URL de asserção é invalida, e ele notifica com um alerta dizendo que isso pode ser uma tentativa de um *spoofing attack*, que é a falsificação de usuário ou serviço para se passar pelo original e roubar dados sigilosos.

- **TS02 -** Este teste visa testar se existe o uso X509, um tipo de certificado digital que viabiliza garantir a integridade tanto do usuário quanto do IdP e dos SP's. Durante a configuração tanto do SP quanto do IdP, é possível notar a utilização de X509 para garantir uma maior integridade e segurança de seus serviços e usuários. No modo *DEBUG* do IdP, ele apresenta a troca de mensagens SAML com o X509 envelopado no SOAP, tanto o certificado quanto a assinatura, ou seja, o teste foi validado com sucesso;

- **TS03 -** Este teste visa viabilizar se existe uma função Hash para proteger a assinatura e o certificado de pessoas que possam vir a interceptar a troca de mensagens entre o IdP e o SP, apesar que o IdP se utiliza de SSL e TLS nas conexões e autenticações para garantir uma maior segurança do usuário na autenticação. É possível ver a utilização de transformações na tabela 14 na tag *<ds:Transforms>* aonde é definido as funções *Hash* que irão digerir a assinatura e o certificado, com isso, este teste está validado com sucesso;

Com relação aos requerimentos dos testes, ao criar um ambiente seguro para troca de mensagens, o IdP utiliza HTTPS com SSL/TLS nos pontos de autenticação. Outro ponto é a utilização de algoritmos no *SAMLResponse* conforme pode ser visto no log da tabela 14, utilizando assinatura envelopada com RSA-SHA1 e com método de digestão em SHA1, ou seja uma função *hash* de dispersão criptográfica que torna as mensagens mais seguras.

Tabela 14 – Log do WSO2 informando o envio do HTTP-POST em HTML form com o token após autenticação - Fonte: Autor

```
[2017-04-22 10:58:42,800] DEBUG
   {org.wso2.carbon.identity.sso.saml.processors.SPInitSSOAuthnRequestProcessor}
   -
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response Destination="http://localhost:8080/sigra/home.jsp"
   ID="_2e4c6cbf7400dedcc20726e9491109f1"
InResponseTo="cjeenbjjicfdaefhkaibopmpgnbkdhhoipghhfbm"
   IssueInstant="2017-04-22T13:58:42.621Z" Version="2.0"
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"><saml2:Issuer
   Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">localhost</saml2:Issuer>
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo><ds:CanonicalizationMethod
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#_2e4c6cbf7400dedcc20726e9491109f1">
<ds:Transforms><ds:Transform
   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:Transform
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/></ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>vCTLOO9vsoW7WoNehOlJ/0YZV54=</ds:DigestValue></ds:Reference>
</ds:SignedInfo><ds:SignatureValue>
[2017-04-22 10:58:42,826] DEBUG
   {org.wso2.carbon.identity.sso.saml.servlet.SAMLSSOProviderServlet} -
<html><body><p>You are now redirected back to
   http://localhost:8080/sigra/home.jsp
If the redirection fails, please click the post button.</p>
<form method='post' action='http://localhost:8080/sigra/home.jsp'>
<p><input type='hidden' name='SAMLResponse' value='TOKEN GIGANTE'>
<button type='submit'>POST</button></p></form><script
   type='text/javascript'>
document.forms[0].submit();</script></body></html>
```

O serviço cadastrado precisa usar um certificado emitido pelo IdP, e também carregar esses dados nos objetos da sessão para que o IdP possa validar o emissor da requisição, logo as aplicações usam um wso2carbon.jks que contém o certificado válido para o IdP. Com a assinatura do *SAMLResponse*, qualquer serviço pode validar o *token* da sessão, assim necessitando de apenas uma autenticação para navegar em múltiplos serviços, ou seja, utilizando-se do mesmo *token* do usuário **pauloesb** tanto no SIGRA quanto no Matrícula Web, para acessar os dois serviços.

Os resultados obtidos nos teste demonstram a importância, no quesito gerencial, de custo, menor manutenção e de alta segurança a funcionalidade em se manter uma

base de dados centralizada a qual o servidor de identidade tem acesso, e qualquer usuário que desejar acessar um recurso no barramento, deverá se autenticar neste servidor com o objetivo de efetivar sua sessão nos serviços disponíveis a ele.

Tabela 15 – Log do WSO2 informando uma possível tentativa de *Spoof Attack* - Fonte: Autor

```
[2017-04-22 12:07:48,861] DEBUG
   {org.wso2.carbon.identity.sso.saml.util.SAMLSSOUtil} -
Request message <?xml version="1.0" encoding="UTF-8"?><samlp:AuthnRequest
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="http://192.168.25.168:8080/matriculaweb/home.jsp"
   Destination="https://192.168.25.168:9443/samlsso"
ForceAuthn="false" ID="edpbelliljkgeekjafgcjkfbdhbnpnkfhakfobla"
   IsPassive="true" IssueInstant="2017-04-22T15:07:38.293Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   Version="2.0"><samlp:Issuer
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:assertion">matriculaweb</samlp:Issuer>
 <saml2p:NameIDPolicy
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" AllowCreate="true"
   Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
SPNameQualifier="Issuer"/>
 <saml2p:RequestedAuthnContext
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" Comparison="exact">
<saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:SAML:2.0:
 assertion">urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
</saml:AuthnContextClassRef></saml2p:RequestedAuthnContext></samlp:AuthnRequest>
[2017-04-22 12:07:48,874] DEBUG
   {org.wso2.carbon.identity.sso.saml.validators.SPInitSSOAuthnRequestValidator}
   - Authentication Request Validation is successful..
[2017-04-22 12:07:48,908] DEBUG
   {org.wso2.carbon.identity.sso.saml.servlet.SAMLSSOProviderServlet} -
   Query string : SAMLRequest= BIG TOKEN
[2017-04-22 12:07:48,908] DEBUG
   {org.wso2.carbon.identity.sso.saml.servlet.SAMLSSOProviderServlet} -
   Unauthenticated User
[2017-04-22 12:07:48,924] ERROR
   {org.wso2.carbon.identity.sso.saml.util.SAMLSSOUtil} - ALERT: Invalid
   Assertion Consumer URL value
'http://192.168.25.168:8080/matriculaweb/home.jsp' in the AuthnRequest
   message from the issuer 'matriculaweb'.
Possibly an attempt for a spoofing attack
```

Os resultados obtidos são satisfatórios, e atendem aos requisitos de segurança relatados pela (ORACLE, 2010), sem contar também que o SAML, por ter sido desenvolvido pela OASIS, e atendendo aos requisitos de segurança da SSTC, está apto ao papel em produção para agir como protocolo de autenticação em uma abordagem centralizada. No contexto referente ao modelo centralizado para garantir a segurança e acesso aos serviços

da UnB, o SAML torna-se um protocolo adequado pois se adequa aos padrões de atender tanto a sistemas legados quanto a sistemas modernos, e com a utilização de um protocolo de transporte SOAP, adaptável a qualquer plataforma.

Cada resultado obtido foi previamente analisado nesta seção e eles juntos conseguem responder as questões de pesquisa: Referente às necessidades de segurança da autenticação em um barramento de serviços, o SAML se provou ser um protocolo robusto que utiliza de diversos parâmetros para aferir a veracidade da asserção assim garantindo uma maior segurança por parte dos autenticados. A cerca da segunda pergunta das questões de pesquisa, o SAML junto ao WSO2 consegue garantir a integridade de autenticação do usuário, assim permitindo o acesso aos serviços disponíveis com apenas uma autenticação, isso pode ser visto graças aos testes de simulação de *Spoof Attack*, sem contar também da utilização do X509 e das funções hash que o próprio algoritmo utiliza.

4 Conclusão

Os sistemas legados são a base de muitas instituições e empresas, e ter um serviço capaz de unificar diferentes sistemas legados, traz um grande benefício de integração e compartilhamento de informações entre plataformas, e sob o contexto evolutivo, a apresentação de uma abordagem centralizada de autenticação reitera a integração entre sistemas legados e modernos com diminuição de custos de manutenção, baixando a complexidade no gerenciamento de informações e aumentando a facilidade de integração com sistemas mais modernos.

Nesse sentido, esta experiência teve como objetivo apresentar a possibilidade de unificar os sistemas de forma segura, atendendo as regras de negócio e permitindo a evolução constante dos sistemas independente se forem sistemas legados ou não. A importância deste estudo para Universidade de Brasília vai se dar uma vez que a UnB conta com inúmeros sistemas legados, e agora com a implantação de um barramento SOA para unificar os sistemas legado, um modelo centralizado de autenticação pode ajudar oferecendo segurança na autenticação e sessões compartilhadas entre múltiplos serviços.

A solução apresentada no modelo centralizado de autenticação atende as necessidades da UnB não somente pelos requisitos de segurança propostos pela Oracle (2010), mas também é um ponto em que o CPD teria de desenvolver em sua solução de integração dos serviços com SOA. Uma abordagem de segurança na autenticação em que centenas de serviços e usuários terão de interagir um com outro, este trabalho demonstra ser uma solução adequada a problemática da falta de integração da autenticação entre os diversos sistemas da UnB.

A validação da proposta por meio de testes com pseudo serviços serviu para demonstrar não somente sua segurança, mas também entender o funcionamento do protocolo SAML que é um protocolo de autenticação robusto. Os testes permitiram investigar, explorar e entender, como funciona o compartilhamento de sessão e os mecanismos de segurança oferecidos pelo SAML. A validação também permitiu responder as questões de pesquisa, ou seja, validando este trabalho de forma a adquirir os argumentos necessários para homologar o protocolo SAML em uma abordagem centralizada. Integridade de identidade e segurança foram os principais tópicos abordados e que com o WSO2 foi possível averiguar a veracidade do uso deles em um ambiente simulado para obter as respostas sobre seu uso com relação ao contexto previamente definido.

Todo o trabalho de desenvolvimento dos pseudo serviços se encontra disponível no repositório do *Git Hub* no seguinte sítio https://github.com/pauloesb/tcc2-unb-fga e pode ser utilizado para acoplar o SAML do WSO2 em qualquer aplicação que têm como

objetivo interagir com o IdP do WSO2. Sendo fácil de configurar, simples de entender e rápido de executar e processar.

Para que um desenvolvedor possa integrar essa solução a sua aplicação, precisar ser realizado um estudo de viabilidade primariamente a depender do tipo do sistema. Nos sistemas mais modernos, são disponibilizados API's de integração como no caso do WSO2 que basta utilizar as classes de implementação do SOAP e do protocolo SAML na sua aplicação, assim removendo a necessidade da autenticação. Nos sistemas legados, talvez isso possa não ser tão trivial pois irá depender se o sistema é totalmente, parcialmente ou independente de plataforma. Para isso a OASIS oferece a documentação para implementação do protocolo SAML, ou seja, os desenvolvedores teriam de desenvolver a camada de interface de comunicação via SOAP e o perfil do SAML desejado, e após esse desenvolvimento, cadastrar a aplicação no IdP conforme esperado, só então a aplicação legado estará conectado ao barramento conforme demonstrado.

4.1 Proposta de Trabalho Futuro

Uma parte importante que seria complementar a este trabalho seria a questão de trabalhar com os protocolos de autorização de usuários aos recursos. O mais usado no mercado atual é o OAuth2, um arcabouço de autorização que permite aplicações de terceiros obter acesso limitado a um serviço HTTP, seja interagindo com o dono do recurso pela orquestração de sua aprovação entre o serviço HTTP eoutro o dono do recurso, ou permitindo que aplicações de terceiros tenham acesso por conta própria (HARDT, 2012).

Um protocolo que foi desenvolvido pela *Sun Microsystems* e padronizado pela OASIS é o *eXtensible Access Control Markup Language* (XACML), baseado em XML e similar ao SAML, é uma linguagem padronizada referente a definições de acesso e controle. O XACML descreve tanto a política de controle e acesso na forma de *tags*, quanto a relação de requisição e resposta que deverão ser aceitas. A *policy language* é usada para expressar políticas de acesso, relacionadas aos acessos e permissões de usuários, aplicações, serviços e recursos (MICROSYSTEMS, 2004).

Dentre esses dois protocolos, o OAuth2 é um recurso *RESTFul* de acordo com o Hardt (2012), diferente do XACML que se utiliza de XML e SOAP para garantir a integridade das autorizações conforme explicado pela Microsystems (2004). A proposta para o trabalho seria integrar um protocolo ou outro ao SAML para garantir não somente a parte de autenticação, mas também a autorização que é considerada essencial em um barramento que possui múltiplos serviços, pois temos a hierarquização de usuários como usuários que só acessam um determinado serviço e não devem ter autorização para acessar a outro serviço e assim por diante.

Referências

AGILAR, E. de V. *Uma Abordagem Orientada a Serviços para a Modernização de Sistemas Legados*. Tese (Doutorado) — Universidade de Brasília, 2016. Citado 7 vezes nas páginas 13, 14, 20, 21, 23, 31 e 35.

ATTANAYAKE, S. *SAML2 Web Browser based SSO with WSO2 Identity Server*. 2010. <http://wso2.com/library/articles/2010/07/saml2-web-browser-based-sso-wso2-identity-server/>. (Accessed on 04/20/2017). Citado na página 39.

BISBAL, J. et al. Legacy information systems: Issues and directions. *IEEE software*, IEEE, v. 16, n. 5, p. 103–111, 1999. Citado 2 vezes nas páginas 18 e 19.

BOX, D. et al. *Simple object access protocol (SOAP) 1.1*. 2000. Citado na página 21.

CARMO, H.; FERREIRA, M. Metodologia da investigação–guia para auto-aprendizagem (2ª edição). *Lisboa: Universidade Aberta*, 2008. Citado na página 15.

COMELLA-DORDA, S. et al. *A survey of legacy system modernization approaches*. [S.l.], 2000. Citado na página 19.

CURBERA, F. et al. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, v. 6, n. 2, p. 86–93, March 2002. ISSN 1089-7801. Citado 2 vezes nas páginas 22 e 23.

DFT. *Digital Certificates Explained - Knowledge Base*. 2015. <https://sites.google.com/site/amitsciscozone/home/security/digital-certificates-explained>. (Accessed on 04/26/2017). Citado na página 41.

GAJASINGHE, K. *[Article] Introduction to Security Assertion Markup Language 2.0*. 2017. <http://wso2.com/library/articles/2014/02/introduction-to-security-assertion-markup-language-2.0/>. (Accessed on 04/20/2017). Citado na página 40.

GROSS, T. Security analysis of the saml single sign-on browser/artifact profile. In: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* [S.l.: s.n.], 2003. p. 298–307. Citado na página 36.

HARDT, D. *RFC 6749 - The OAuth 2.0 Authorization Framework*. 2012. <https://tools.ietf.org/html/rfc6749>. (Accessed on 04/22/2017). Citado na página 58.

KRAFZIG, D. et al. *Service-Oriented Architecture Best Practices (The Coad Series)*. [S.l.]: Prentice Hall PTR, Upper Saddle River, NJ, 2004. Citado 2 vezes nas páginas 20 e 21.

LOPEZ, J.; OPPLIGER, R.; PERNUL, G. Authentication and authorization infrastructures (aais): a comparative survey. *Computers & Security*, Elsevier, v. 23, n. 7, p. 578–590, 2004. Citado na página 24.

MALDONADO, J. C. et al. Introduçao ao teste de software. *Rio de Janeiro: Campus*, 2007. Citado na página 48.

MICROSYSTEMS, S. *Sun's XACML Implementation*. 2004. <http://sunxacml.sourceforge.net/>. (Accessed on 04/22/2017). Citado na página 58.

MUMBAIKAR, S.; PADIYA, P. et al. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, v. 3, n. 5, 2013. Citado 3 vezes nas páginas 23, 24 e 35.

ORACLE. *Understanding Web Services Security Concepts*. 2010. <https://docs.oracle.com/cd/E17904_01/web.1111/b32511/intro_security.htmWSSEC2344>. (Accessed on 04/21/2017). Citado 4 vezes nas páginas 49, 50, 55 e 57.

PAPAZOGLOU, M. P. Service-oriented computing: concepts, characteristics and directions. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.* [S.l.: s.n.], 2003. p. 3–12. Citado na página 19.

PINTO, H. L. M.; BRAGA, J. L. Sistemas legados e as novas tecnologias: técnicas de integração e estudo de caso. *Informática Pública, Belo Horizonte*, v. 7, n. 1, p. 48–69, 2004. Citado 3 vezes nas páginas 12, 13 e 19.

RAGOUZIS, N. et al. Oasis security assertion markup language (saml) v2. 0 technical overview. *Committee Draft (25 March 2008) http://www.oasis-open.org/committees/download.php*, v. 27819, 2008. Citado 11 vezes nas páginas 25, 26, 28, 29, 30, 31, 32, 33, 34, 35 e 36.

RIOS, E.; MOREIRA, T. *Teste de software*. [S.l.]: Alta Books Editora, 2006. Citado 2 vezes nas páginas 47 e 49.

SNEED, H. M. Integrating legacy software into a service oriented architecture. In: *Conference on Software Maintenance and Reengineering (CSMR'06)*. [S.l.: s.n.], 2006. p. 11 pp.–14. ISSN 1534-5351. Citado na página 19.

SUDA, B. Soap web services. *Retrieved June*, v. 29, p. 2010, 2003. Citado 3 vezes nas páginas 21, 22 e 23.

WALEED, G. M.; AHMAD, R. B. Security protection using simple object access protocol (soap) messages techniques. In: *2008 International Conference on Electronic Design*. [S.l.: s.n.], 2008. p. 1–6. Citado na página 36.

WEIDERMAN, N. H. et al. *Approaches to Legacy System Evolution*. [S.l.], 1997. Citado na página 12.

Apêndices

APÊNDICE A – Código fonte do pseudo serviço

A.1 GerenciamentoContexto.java

```
package matriculaweb;

import org.wso2.carbon.identity.sso.agent.SSOAgentConstants;
import org.wso2.carbon.identity.sso.agent.bean.SSOAgentConfig;
import org.wso2.carbon.identity.sso.agent.SSOAgentException;
import org.wso2.carbon.identity.sso.agent.saml.SSOAgentX509Credential;
import org.wso2.carbon.identity.sso.agent.saml.SSOAgentX509KeyStoreCredential;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;

public class GerenciamentoContexto implements ServletContextListener {

   private final static String PATH = "matriculaweb";
   private final static String PATH_PROPERTIES =
       "/WEB-INF/classes/matriculaweb.properties";
   private final static String PATH_CERT = "/WEB-INF/classes/wso2carbon.jks";
   private final static String PROPERTIE_KSP = "KeyStorePassword";
   private final static String PROPERTIE_IDPCERTALIAS = "IdPPublicCertAlias";
   private final static String PROPERTIE_PKA = "PrivateKeyAlias";
   private final static String PROPERTIE_PKP = "PrivateKeyPassword";

   public final static String PROPERTIE_SAML2IDPURL = "SAML2.IdPURL";

    private static Logger logger =
        Logger.getLogger(GerenciamentoContexto.class.getName());
    private static Properties propriedades = new Properties();
```

```
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        try {
            if(servletContextEvent.getServletContext().getContextPath().contains(PATH))
                {
                propriedades.load(servletContextEvent.getServletContext().
                getResourceAsStream(PATH_PROPERTIES));
            }
            SSOAgentConfig config = new SSOAgentConfig();
            config.initConfig(propriedades);
            config.getSAML2().setSSOAgentX509Credential(
            associandoCredencial(servletContextEvent.getServletContext().
                  getResourceAsStream(PATH_CERT), propriedades));
            servletContextEvent.getServletContext().setAttribute(
            SSOAgentConstants.CONFIG_BEAN_NAME, config);
        } catch (IOException e){
            logger.log(Level.SEVERE, e.getMessage(), e);
        } catch (SSOAgentException e) {
            logger.log(Level.SEVERE, e.getMessage(), e);
        }
    }

    public SSOAgentX509Credential associandoCredencial(InputStream
        keyStoreInputStream, Properties propriedades) throws SSOAgentException {
        return new SSOAgentX509KeyStoreCredential(keyStoreInputStream,
                        propriedades.getProperty(PROPERTIE_KSP).toCharArray(),
                        propriedades.getProperty(PROPERTIE_IDPCERTALIAS),
                        propriedades.getProperty(PROPERTIE_PKA),
                        propriedades.getProperty(PROPERTIE_PKP).toCharArray());

    }

    public void contextDestroyed(ServletContextEvent servletContextEvent) {

    }

    public static Properties getProperties(){
        return propriedades;
    }
}
```

A.2 FiltroSAMLSSO.java

```
package matriculaweb;

import org.apache.axiom.om.util.Base64;
import org.apache.commons.lang.StringUtils;
import org.wso2.carbon.identity.sso.agent.SSOAgentConstants;
import org.wso2.carbon.identity.sso.agent.SSOAgentFilter;
import org.wso2.carbon.identity.sso.agent.bean.SSOAgentConfig;

import javax.servlet.*;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;

public class FiltroSAMLSSO extends SSOAgentFilter {

    private static Logger LOGGER =
        Logger.getLogger(FiltroSAMLSSO.class.getName());

    private static final String HTTP_POST_BIND =
        "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST";
    private static final String USUARIO = "username";
    private static final String SENHA = "password";
    private static final String ENCODE = "UTF-8";
    private static final String HTML_A = "<html>\n" +
                                "<body>\n" +
                                "<form method='post' action='";
    private static final String HTML_B = "'>\n" +
                                 "<input type='hidden' name='sectoken' value='";
    private static final String HTML_C = "'/>\n" +
                                 "</form>\n" +
                                 "<script type='text/javascript'>\n" +
                                 "document.forms[0].submit();\n" +
                                 "</script>\n" +
                                 "</body>\n" +
                                 "</html>";

    private static Properties propriedades =
        GerenciamentoContexto.getProperties();
```

```
protected FilterConfig filtroConfig = null;

@Override
public void init(FilterConfig filterConfig) throws ServletException {
    this.filtroConfig = filterConfig;
}

@Override
public void doFilter(ServletRequest servletRequest, ServletResponse
    servletResponse,
                    FilterChain filterChain) throws IOException,
                        ServletException {

    String binding = servletRequest.getParameter(
    SSOAgentConstants.SSOAgentConfig.SAML2.HTTP_BINDING);
    if(binding == null || binding.isEmpty() ||
        !("HTTP-POST".equals(binding))){
     binding = HTTP_POST_BIND;
    }
    SSOAgentConfig config =
        (SSOAgentConfig)filtroConfig.getServletContext().
    getAttribute(SSOAgentConstants.CONFIG_BEAN_NAME);
    config.getSAML2().setHttpBinding(binding);
    redirecionamento(config, servletRequest);
    servletRequest.setAttribute(SSOAgentConstants.CONFIG_BEAN_NAME,config);
    super.doFilter(servletRequest, servletResponse, filterChain);
}

public String postHtml(String usuario, String senha, String samlidpurl)
    throws UnsupportedEncodingException {
  String autorizacao = usuario + ":" + senha;
  autorizacao = new String(Base64.encode(autorizacao.getBytes(ENCODE)));
  return HTML_A + samlidpurl + HTML_B + autorizacao + HTML_C;
}

public void redirecionamento(SSOAgentConfig config, ServletRequest
    request) {
  if (StringUtils.isNotEmpty(request.getParameter(USUARIO)) &&
          StringUtils.isNotEmpty(request.getParameter(SENHA))) {
     try {
        config.getSAML2().setPostBindingRequestHTMLPayload(postHtml(
             request.getParameter(USUARIO),
```

```
                request.getParameter(SENHA),
                propriedades.getProperty(
                  GerenciamentoContexto.PROPERTIE_SAML2IDPURL)));
         } catch (UnsupportedEncodingException e) {
            LOGGER.log(Level.SEVERE, e.getMessage());
         }
        } else {
            config.getSAML2().setPostBindingRequestHTMLPayload(null);
        }
    }

    @Override
    public void destroy() {

    }
}
```

A.3 RedirecionamentoServlet.java

```
package matriculaweb;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class RedirecionamentoServlet extends HttpServlet {

   private final static String SAML_URI = "/samlsso";
   private final static String HOME_URI = "home.jsp";
   private final static String LOGOUT_URI = "/logout";
   private final static String INDEX_URI = "index.jsp";

    protected void service(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        if(request.getRequestURI().endsWith(SAML_URI)){
            request.getRequestDispatcher(HOME_URI).forward(request,response);
        } else if (request.getRequestURI().endsWith(LOGOUT_URI)){
            request.getRequestDispatcher(INDEX_URI).forward(request,response);
```

```
        }
    }

}
```

A.4 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
    version="2.5">
   <display-name>Matricula Web</display-name>

    <servlet>
        <servlet-name>RedirecionamentoServlet</servlet-name>
        <servlet-class>matriculaweb.RedirecionamentoServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>RedirecionamentoServlet</servlet-name>
        <url-pattern>/samlsso</url-pattern>
        <url-pattern>/logout</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>FiltroSAMLSSO</filter-name>
        <filter-class>matriculaweb.FiltroSAMLSSO</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>FiltroSAMLSSO</filter-name>
        <url-pattern>*.jsp</url-pattern>
        <url-pattern>/samlsso</url-pattern>
        <url-pattern>/logout</url-pattern>
    </filter-mapping>

    <listener>
        <listener-class>matriculaweb.GerenciamentoContexto</listener-class>
    </listener>
```

```
    <listener>
        <listener-class>
        org.wso2.carbon.identity.sso.agent.saml.SSOAgentHttpSessionListener
        </listener-class>
    </listener>

    <error-page>
        <exception-type>
        org.wso2.carbon.identity.sso.agent.SSOAgentException
        </exception-type>
        <location>/errors.jsp</location>
    </error-page>
</web-app>
```

A.5 home.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="java.util.Map" %>
<%@ page import="java.util.Iterator" %>
<%@ page import="org.wso2.carbon.identity.sso.agent.bean.LoggedInSessionBean"
    %>
<%@ page import="org.wso2.carbon.identity.sso.agent.bean.SSOAgentConfig" %>
<%@ page import="org.wso2.carbon.identity.sso.agent.SSOAgentConstants" %>
<%@ page contentType="text/html;charset=UTF-8" language="java"
    pageEncoding="UTF-8" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <title>Matricula Web Logado</title>
</head>
<%String identificador = null;%>
<%if(request.getSession(false) != null &&
   request.getSession(false).getAttribute(SSOAgentConstants.SESSION_BEAN_NAME)
       == null){%>
        <%request.getSession().invalidate();%>
        <script type="text/javascript">
        location.href = "index.jsp";
        </script>
```

```
<%return;}%>

<%SSOAgentConfig ssoAgentConfig =
    (SSOAgentConfig)getServletContext().getAttribute(
SSOAgentConstants.CONFIG_BEAN_NAME);%>
<%LoggedInSessionBean sessaoBean =
    (LoggedInSessionBean)session.getAttribute(SSOAgentConstants.SESSION_BEAN_NAME);%>
<%LoggedInSessionBean.AccessTokenResponseBean accessTokenResponseBean = null;%>

<%if(sessaoBean != null){%>
   <%if(sessaoBean.getSAML2SSO() != null) {%>
      <%identificador = sessaoBean.getSAML2SSO().getSubjectId();%>
        <%accessTokenResponseBean =
            sessaoBean.getSAML2SSO().getAccessTokenResponseBean();%>
    <%} else {%>
         <script type="text/javascript">
              location.href = "index.jsp";
           </script>
         <%return;}%>
   <%} else {%>
        <script type="text/javascript">
            location.href = "index.jsp";
        </script>
  <%return;}%>
<body>
   <div>
      <div class="login-form">
       <%if(identificador != null &&
           ssoAgentConfig.getSAML2().isSLOEnabled()){%>
         <h1>Sistema Matricula Web</h1><br/>
           <h2> Voce esta logado como: <%=identificador%></h2>
           <a type="submit" class="log-btn"
               href="logout?SAML2.HTTPBinding=HTTP-POST">Logout (HTTP
               Post)</a><br/>
       <%}%>
      </div>
   </div>
   <script
      src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>
   <script src="js/index.js"></script>
</body>
</html>
```

A.6 index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ page import="org.wso2.carbon.identity.sso.agent.bean.SSOAgentConfig"%>
<%@ page import="org.wso2.carbon.identity.sso.agent.SSOAgentConstants"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link rel="stylesheet" type="text/css" href="css/style.css">
<title>Login Matricula Web</title>
</head>
<body>
<div class="login-form">
     <h1>Matricula Web</h1>
     <form method="post" action="samlsso?SAML2.HTTPBinding=HTTP-POST">
     <div class="form-group ">
       <input type="text" class="form-control" placeholder="Usuario"
           id="username">
       <i class="fa fa-user"></i>
     </div>
     <div class="form-group log-status">
       <input type="password" class="form-control" placeholder="Senha"
           id="password">
       <i class="fa fa-lock"></i>
     </div>
      <span class="alert">Invalid Credentials</span>
     <input type="submit" class="log-btn" value="Login"/>
     </form>
</div>
<script
    src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>
<script src="js/index.js"></script>
</body>
</html>
```

Anexos

ANEXO A – Troca de mensagem do WSO2 entre IdP e SP

A.1 SAMLResponse com Redirecionamento

Tabela 16 – Mensagem *SAMLResponse* com *Token* e redirecionamento - Fonte: Autor

```
<html>
    <body>
                <p>You are now redirected back to
                    http://localhost:8080/sigra/home.jsp
                If the redirection fails, please click the post button.</p>
                <form method='post'
                    action='http://localhost:8080/sigra/home.jsp'>
                    <p>
                    <input type='hidden' name='SAMLResponse' value='BIG
                        TOKEN'>
                    <button type='submit'>POST</button>

                    </p>
                </form>
                <script type='text/javascript'>
                    document.forms[0].submit();
                </script>
    </body>
</html>
```

A.2 Authnrequest com Assinatura

Tabela 17 – Mensagem *AuthnRequest* com assinatura para requisição de recurso - Parte 1 - Fonte: Autor

```
<?xml version="1.0" enconding="UTF-8"?>
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="http://localhost:8080/sigra/home.jsp"
Destination="https://localhost:9443/samlsso" ForceAuthn="false"
ID="joegfekjhmheklmdgibeifnlkgnnjnabmpfcicnc" IsPassive="false"
IssueInstant="2017-04-16T01:03:00:363Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Version="2.0">
```

Tabela 18 – Mensagem *AuthnRequest* com assinatura para requisição de recurso - Parte 2 - Fonte: Autor

```
        <samlp:Issuer xmlns:samlp="urnoasis:names:tc:SAML:2.0:assertion">
    sigra</samlp:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc/c14n#"/>
            <ds:SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="joegfekjhmheklmdgibeifnlkgnnjnabmpfcicnc">
                <ds:Transforms>
                    <ds:Transform Algorithm=
                    "http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>FIIQ2bYwxR0gNZLC8D/1/0gUwN4</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>BIG SIGNED VALUE</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>BIG X509 Certificate
                    VALUE</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </Signature>
    <saml2p:NameIDPolicy
        xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
        AllowCreate="true"
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
        SPNameQualifier="Issuer" />
    <saml2p:RequestedAuthnContext
        xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
        Comparison="exact">
        <saml:AuthnContextClassRef
            xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
            urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
    </saml2p:RequestedAuthnContext>
</samlp:AuthnRequest>
```

A.3 SAMLResponse Autorizado e Assinado

Tabela 19 – Mensagem *SAMLResponse* com autorização ao recurso - Parte 1 - Fonte: Autor

```
<?xml version="1.0" enconding="UTF-8"?>
<saml2p:Response Destination="http://localhost:8080/sigra/home.jsp"
ID="_9ddb5c819190d84b86946a1c9108bed9"
    InResponseTo="pccomegnpknhpopkcingkdjnajldbgjbjlnmbohcp"
IssueInstant="2017-04-20T19:41:35.067Z" Version="2.0"
    xmlns:"urn:oasis:names:tc:SAML:2.0:protocol">
    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">localhost</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="_9ddb5c819190d84b86946a1c9108bed9">
                <ds:Transforms>
                    <ds:Transform Algorithm=
                    "http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>QBZ0g3y9+RUS4VzA+0rgaLGNbkg-</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>BIG SIGNATURE VALUE</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate> BIG X509 CERTIFICATE
                    VALUE</ds:X509Certificate>
            </ds:X509>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2p:Status>
        <saml2p:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </saml2p:Status>
    <saml2:Assertion ID="_f99bd041e0a1a33fbefffb2e8bc00bc8"
        IssueInstant="2017-04-20T19:41:35.074Z"
    Version="2.0" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
```

Tabela 20 – Mensagem *SAMLResponse* com autorização ao recurso - Parte 2 - Fonte: Autor

```
            <saml2:Issuer
                Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
            xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">localhost</saml2:Issuer>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
                        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <ds:SignatureMethod
                        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="_9ddb5c819190d84b86946a1c9108bed9">
                        <ds:Transforms>
                            <ds:Transform Algorithm=
                            "http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                            <ds:Transform
                                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
                            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>N3SbHOKsfkqANln9iNGDOYDOkS4=</ds:DigestValue>
                    </ds:Referece>
                </ds:SignedInfo>
                <ds:SignatureValue>BIG SIGNATURE VALUE</ds:SignatureValue>
                <ds:KeyInfo>
                    <ds:X509Data>
                        <ds:X509Certificate>BIG X509 CERTIFICATE
                            VALUE</ds:X509Certificate>
                    <ds:X509Data>
                </ds:KeyInfo>
            </ds:Signature>
        <saml2:Subject>
            <saml2:NameID
                Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
            admin</saml2:NameID>
            <saml2:SubjectConfirmation
                Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml2:SubjectConfirmationData
                    InResponseTo="pcc0megnpknhpopkcingkdjnajlbgjbjlnmbohcp"
                NotOnOrAfter="2016-04-20T19:46:35.067Z"
                    Recipient="http://localhost:8080/sigra/home.jsp"/>
            </saml2:SubjectConfirmation>
        </saml2:Subject>
        <saml2:Conditions NotBefore="2017-04-20T19:41:35.074Z"
            NotONOrAfter="2017-04-20T19:46:35.067Z">
            <saml2:AudienceRestriction>
```

Tabela 21 – Mensagem *SAMLResponse* com autorização ao recurso - Parte 3 - Fonte: Autor

```
            </saml2:AudienceRestriction>
        </saml2:Conditions>
        <saml2:AuthnStatement AuthnInstant="2017-04-20T19:41:35.110Z"
            SessionIndex="49193561-8018-45b0-9240-26fa8ea8f5f5"
            <saml2:AuthnContext>
                <saml2:AuthnContextClassRef>
                    urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtected
                </saml2:AuthnContextClassRef>
            </saml2:AuthnContext>
        </saml2:AuthnStatement>
    </saml2:Assertion>
    </saml2p:Response>
```

A.4 Telas dos Pseudo-Serviços e da Autenticação

Figura 16 – Tela de login do pseudo serviço index.jsp - Fonte: Autor

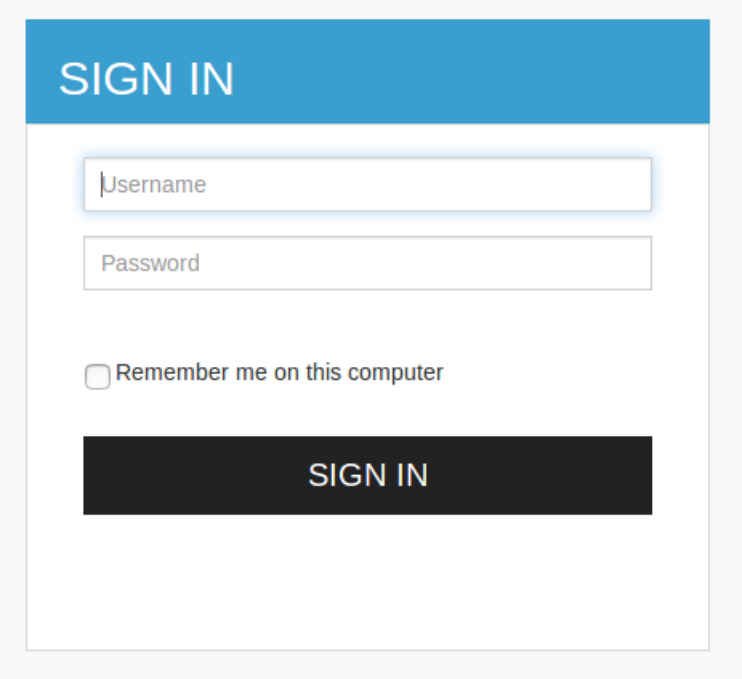

Figura 17 – Tela de login do IdP WSO2 após redirecionamento para o SSO SAML - Fonte: Autor

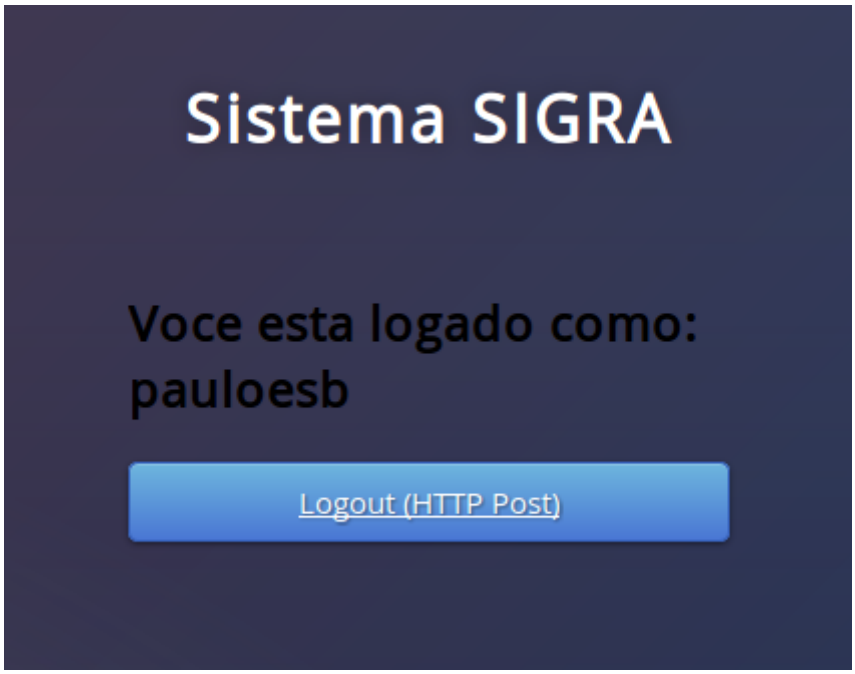

Figura 18 – Tela de pós-login home.jsp do SIGRA - Fonte: Autor

Figura 19 – Tela de pós-login home.jsp do Matrícula Web - Fonte: Autor